

GIMA

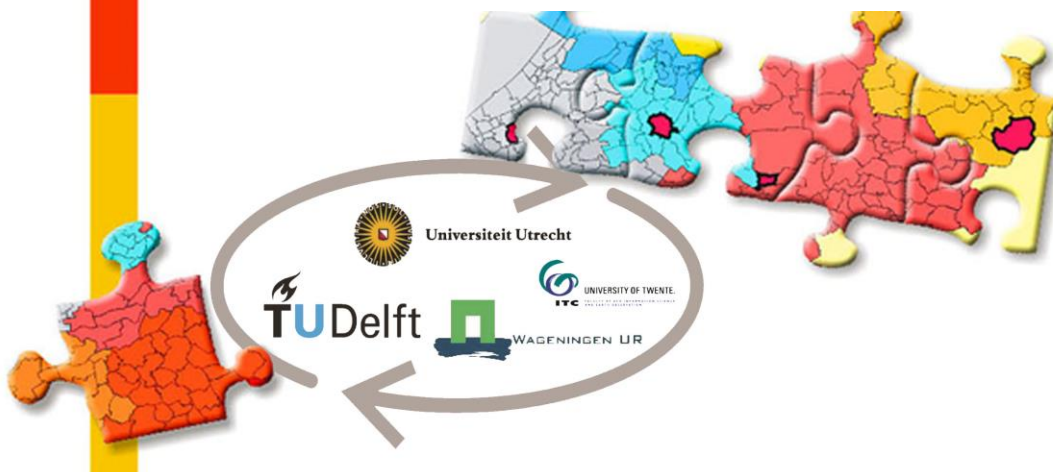
Geographical Information Management and Applications

Background extraction from videos produced in laboratory experiments

MSc Thesis
December 2013

Cecilia Herrera

Professor: Prof. dr. Massimo Menenti
Supervisor: Dr. ir. Ben Gorte



Background extraction from videos produced in laboratory experiments

Thesis submitted in partial fulfillment of the degree

Master of Science

in

Geographical Information Management and Applications

December 2013

Author: Cecilia Herrera

Professor: Prof. dr. Massimo Menenti (Delft University of Technology)

Supervisor: Dr. ir. Ben Gorte (Delft University of Technology)

Coordinator: Dr. dipl. ing. Sisi Zlatanova (Delft University of Technology)

DISCLAIMER

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my supervisors.

I understand that my thesis may be made electronically available to the public.

Abstract

In laboratory cages or arenas, animals are tracked to study their behavior, movement and activity. But before animals can be tracked, they first have to be detected separate from the background. In order to do this, the background without the animal (e.g. “empty” or reference background) needs to be available. This background is sometimes easy to obtain, i.e. when the animal is hidden or absent. Other times, it has to be extracted from sequences of images while the animal is present. This involves estimating the background in the places where the animal is located over time.

There are different methods or algorithms to extract the background. One of these is the background modeling component of the so-called “Background Subtraction” methods, which are commonly used to extract moving objects from video sequences. The objective of this study was to find the algorithm among a set of different algorithms that best provides the empty background for videos produced in laboratory experiments. To achieve this, the requirements of a good algorithm were identified and different algorithms were studied. An evaluation method to derive the best algorithm was devised. The evaluation was then performed and the best algorithm was identified. The chosen algorithm was thereafter tested to see how it performed using videos showing different situations in the laboratory.

The identified requirements of a good algorithm were high speed, good quality of the extracted background and applicability to different situations in the laboratory. A total of 24 algorithms were examined. The identified measures used to evaluate the algorithms were (1) Speed or the time it takes to run the algorithm, (2) Minimum Thresholded Difference, which is calculated as the lowest “thresholded” difference between the extracted and reference background images, taken over all the frames in the video, and (3) the Frequency by which this lowest difference value was calculated in the whole video. The algorithms were run with 2 sample videos and the values for the 3 measures were obtained. The multi-criteria evaluation (MCE) procedure was then used with the extracted values to rank the different algorithms. The best algorithm found was the LBMixtureOfGaussians. This algorithm was run with 37 videos showing different situations in the laboratory, and empty backgrounds were extracted in most of the cases. Only in videos where the animal hardly moved or in multiple arenas where more than one animal was present, were empty backgrounds not obtained.

There were some results which were not as expected. For example, the algorithms FrameDifference, StaticFrameDifference, and AdaptiveBackgroundLearning, which always produced backgrounds with animal or traces of animal, got relatively high scores in the MCE analysis. Reasons for this behavior were sought and recommendations were made. It was also suggested to optimize the extraction of the background.

Keywords:

Background extraction, background subtraction, background modeling, empty background, reference background, minimum (thresholded) difference, Mixture of Gaussians algorithm, Multi-criteria evaluation, videos, laboratory experiments.

Acknowledgements

I would like to thank my husband and children for bearing with me while I spend many hours and nights alone in the room to study. Thanks for the support and exchange of ideas on how I should proceed with the thesis and my study.

I also want to thank my parents, brothers and sisters who are always there to support me in all aspects. I am sorry that I could not spend time with you as much as I wanted because I was busy with my study.

I would like to acknowledge Wil van Dommelen, my supervisor at Noldus Information Technology, b.v., for giving me time and allowing me to use resources at the office for my thesis. I know that the topic of my thesis is relevant to our work in the office.

I would like to thank my thesis supervisors Massimo Menenti and Ben Gorte, as well as the Module 8 coordinator Sisi Zlatanova for their support and patience in dealing with me, and for encouraging me to improve my thesis.

To the rest of the GIMA staff involved over the past three years, thank you for providing the necessary background and foundation in our course work leading up to this thesis. I am glad that the GIMA curriculum is structured so that working people like me could also participate.

And last but not the least, I would like to thank my fellow part- and full time students for their company and friendship during the study. I have enjoyed the time I had with you, especially the dinners, drinks and walks we had during the contact weeks.

Cecilia Herrera
Ede, 2013

Table of Contents

Abstract	i
Acknowledgements	ii
List of tables	iv
List of figures	iv
I. Introduction	1
Background Subtraction	2
Background subtraction algorithms	2
Challenges to background modeling	3
Objectives.....	4
Scope of the research.....	5
II. Literature Review	6
A. Background representation	6
A.1. Type of method	6
A.2. Classification based on features	9
B. Background initialization	9
C. Background adaptation	10
Other considerations.....	11
Background extraction algorithms	11
Evaluation method.....	12
III. Methodology.....	15
Find a suitable background extraction algorithm – Which is the best algorithm?.....	15
1. Which algorithms can be tested	15
2. Which quantitative and reproducible measures	15
3. How can the evaluation be performed	16
4. How does the algorithm perform	17
IV. Results and Discussion.....	19
Find a suitable background extraction algorithm – Which is the best algorithm?.....	19
1. Which algorithms can be tested	19
2. Which quantitative and reproducible measures	21
3. How can the evaluation be performed	27
4. How does the algorithm perform	37
V. Summary and Conclusions	39
References	41
Appendix I. Algorithms used to extract the background.....	44
Appendix II. Comparison of GrimsonGMM and LBMOG using different videos.	47
Appendix III. Frames corresponding to quality measures for different background extraction methods using BS2 video.	50
Appendix IV. Multi-criteria evaluation to select the best method using BS2 video with thresholded Difference values.....	52
Appendix V. Frames corresponding to quality measures for different background extraction methods using Video1.	54
Appendix VI. Multi-criteria evaluation to select the best method using Video1 video with thresholded Difference values.	56
Appendix VII. Extracted backgrounds using LBMOG.....	58

List of tables

Table 1. Speed (msec) results for LBMOG and GrimsonGMM using 500 samples.....	23
Table 2. Minimum and maximum values for runs with BS2, sampling interval 1 (Number represents the frequency of the same value of minimum Difference.	24
Table 3. Methods which produced at least one empty background for BS2 video.	28
Table 4. Summary of calculated values for the different criteria for the BS2 video.	29
Table 5. Frequency values of minimum threshold difference plus different number of pixels, using BS2 video.	32
Table 6. Top ranking algorithms for different scenarios and frequency values (minimum thresholded difference plus the factor), for BS2 video.	32
Table 7. Summary of calculated values for the different criteria for the Video1 video.....	34
Table 8. Frequency values of minimum threshold difference plus different number of pixels, using Video1 video.	36
Table 9. Top ranking algorithms for different scenarios and frequency values (minimum thresholded difference plus the factor), for Video1 video.	36

List of figures

Figure 1. Flow diagram of a generic background subtraction algorithm (after Cheung and Kamath, 2003)	2
Figure 2. CPU, Memory and Time Consumption of Background Subtraction algorithms in BGSLibrary (from Sobral, 2013).	13
Figure 3. Steps undertaken in the methodology.....	18
Figure 4. Speed results for LBMOG and GrimsonGMM using 500 samples.....	24
Figure 5. Difference values for LBMOG using video BS2, sampling interval 1, (normal values in blue, left axis; thresholded values in red, right axis).	25
Figure 6. Difference values for GrimsonGMM using video BS2, sampling interval 1 (normal values in blue, left axis; thresholded values in red, right axis).....	25
Figure 7. LBMOG reference and resulting images for BS2, sampling rate 1.....	26
Figure 8. GrimsonGMM reference and resulting images for BS2, sampling rate 1.....	26
Figure 9. Relation between Speed results with Type of algorithm (left axis) and Modality (right axis) for BS2 video.....	30
Figure 10. Relation between Quality results with Type of algorithm (left axis) and Modality (right axis) for BS2 video.....	30
Figure 11. Relation between Speed results with Type of algorithm (left axis) and Modality (right axis) for Video1 video.....	34
Figure 12. Relation between Quality results with Type of algorithm (left axis) and Modality (right axis) for Video1 video.....	35

I. Introduction

Computer applications that locate and track moving objects are popular nowadays. These applications are normally used to study what the objects do. Some applications make use of sensors, like the Global Positioning System (GPS) for locating the objects. Others make use of images taken live or from videos.

An example of such applications is software which is used in experiments to track animals placed in cages in the laboratory. Videos of the animals are taken and studied with the software. One requirement to do the tracking is to be able to distinguish the background from the moving objects (also called the foreground objects), i.e. the animals. When this is possible, the background can be extracted or removed from the images and only the animals will be seen. The extracted background is considered an “empty background”.

The empty background is sometimes easy to obtain, i.e. when foreground object(s) is hidden or absent. At other times, the background has to be extracted using “cluttered sequences” where parts of the background are obstructed (Reddy, et al., 2009). Background extraction involves detecting where changes took place between two images, and estimating the background in the places where the changes took place. Here, the changes in the images are assumed to correspond to the location of the foreground object. This thesis was undertaken to find a method that will best provide the empty background for videos produced in laboratory experiments. There have been many studies in background extraction applied in different domains, but none was found to have been applied in different laboratory situations.

There are various methods to get the empty background. They work on pixel-level, region-level, or hybrid of the two. Examples of pixel methods are image inpainting (Criminisi, et al., 2004), median filter, pixels of stable intensity, use of codewords and clustering (Reddy, et al., 2011). These methods can perform well, but can suffer from degraded visual quality and can fail when the time interval of exposure of the foreground object is more than that of the background. Region or block level methods (Reddy, et al., 2011) can use temporal sum of absolute differences, clustered blocks, Markov Random Fields or motion parameters (Varadarajan, et al., 2009). They can result in errors if objects are quite stationary for extended periods, can have problems with blending of foreground and backgrounds, can be quite complex and can fail when the background is dynamic (e.g. waving trees). Hybrid approaches (Reddy, et al., 2011) can make use of motion information and energy function with data and smoothness terms, but can be computationally quite complex.

Other background extraction methods have been proposed, some of which do simultaneous foreground tracking and background extraction and updating (Varadarajan, et al., 2009). These methods were criticized before as being computationally very expensive. However, with the capacity of computers nowadays, this is becoming less and less of a problem. This study examines the so-called “Background Subtraction” methods, which are commonly used to extract moving objects from video sequences. These methods contain components which model the background. The background modeling part of these methods will be examined in this study. They will be tested to see if one can be used to properly extract the background for laboratory situations.

Following is a general introduction to background subtraction concept and its applications. Then the background modeling part will be discussed in more detail.

Background Subtraction

For many years, the method of locating and tracking objects in video frames has been to use (1) an image of the stationary background, (2) a second image with animals or moving objects and (3) consider the difference of the two images (e.g. the parts that have significantly changed) as the animal or moving object (Freedman and Russel, 1997). This process is called “background subtraction”, and the stationary background is called the background model (Elgammal, 2011). Background subtraction is the basis (Reddy, et al., 2011) of a “quick and dirty” (Benezeth, et al. 2008) way of detecting moving objects. It is a “computationally affordable” method for real time applications (Maddalena and Petrosino, 2008).

The roots of background subtraction date back to photography in the 19th century when the background image was obtained by allowing exposure of the film to continue for a period of time longer than needed for moving objects to traverse the field of view (Freedman and Russel, 1997). The technique has been used in Landsat Imagery by Eghbali (1979) where he successfully isolated regions of change using the absolute value of maximum difference. In 1979, Jain and Nagel, compared the first frame with the second and subsequent frames of a TV-image sequence to detect moving objects. The concept of background subtraction was also used in early human motion analysis systems (Elgammal, 2011), multimedia applications (El Baf, et al., 2008b) and video compression (Maddalena and Petrosino, 2009). Another important application of background subtraction is video surveillance. Humans or vehicles are observed in real time to provide a description about the activities of the objects with respect to the environment and among themselves (Chen, et al., 2005). Example environments are banks, shopping malls, airports, train stations and roads. Animals, like pigs, mice and rats have also been the subject of video surveillance (McFarlane and Schofield, 1995 and Noldus, et al. 2001). Video surveillance is also related to video analysis (e.g. in sports) and motion capture applications.

Background subtraction algorithms

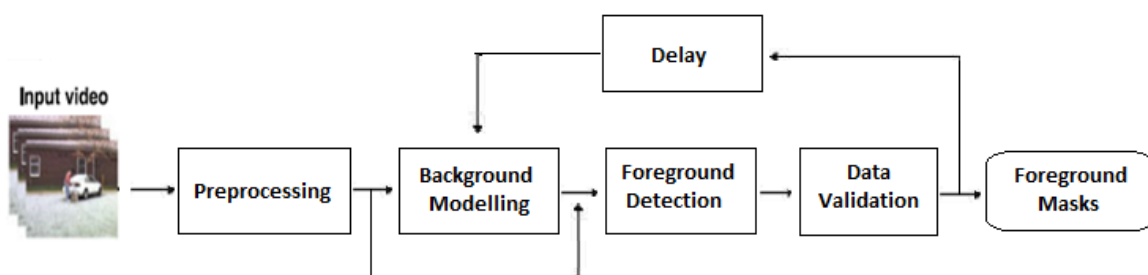


Figure 1. Flow diagram of a generic background subtraction algorithm (after Cheung and Kamath, 2003)

Most background subtraction algorithms follow a flow diagram as shown in Figure 1. Preprocessing changes the raw input video into a format that can be processed. This can involve applying temporal and/or spatial smoothing to reduce camera noise. For real-time systems, frame-size and frame-rate are reduced to lower data processing rate. Image registration is employed when the camera is moving or multiple cameras are involved. The choice of the feature type (intensity or color in HSV or

RGB) is also made in this step. Background modeling uses each new video frame to calculate and update the background model. If a good background is already available, this step could be skipped. Foreground detection identifies pixels in the frame that are not classified as background and outputs them as candidates of the foreground mask. Data validation examines the candidate foreground masks and eliminates the pixels that do not correspond to actual moving objects. After this, the algorithm can update the background model again, causing a delay, until data validation is sufficient. The final output is the foreground mask.

The general requirements (Elhabian, et al., 2008) for background subtraction algorithms are accuracy in object contour detection and temporal stability. Sensitivity to changes of small magnitude and good accuracy under varying conditions is also desired.

The background modeling activity, where the background is extracted and updated, is the main interest in this study.

Challenges to background modeling

There are some well-known issues or challenges in background modeling (Madalena and Petrosino, 2008, Elgammal, 2011). They are caused by changes that occur to the background over time. The changes can affect only part of the background (local), or can affect the entire background (global). The background model should tolerate these changes by being invariant or adapting to them. The following lists the background changes that can occur, the conditions in the laboratory, and how the changes are dealt with.

1. *light changes*: gradual illumination changes could be due to change in the relative location of the sun during the day, whereas sudden illumination changes could be due to switching the lights on or off or changing between sunny or cloudy conditions. Indoor laboratory experiments do not suffer from gradual illumination changes due to sun or weather conditions, but can be affected by sudden changes due to lights switching on or off. To prevent problems with sudden changes, the period to be considered for running the background extraction algorithm can be limited to only the part where no sudden changes in lighting occurs.
2. *moving background*: global changes could be due to camera displacements, whereas local changes could be due to moving background objects like tree branches or rippling water due to wind. Most laboratory experiments do not suffer much from moving background, since the camera position is not changed (remains stationary) during the course of the experiment, and there are no external moving objects like trees. However, the video can:
 - a. contain parts where the laboratory technician changes his own position, moves objects outside the cage, or moves the cages, and
 - b. the animal in the cage can also move and affect its surroundings like moving straw bedding or creating ripples in the water.

For (a), the video frames can be masked so that only the part inside the arena will be considered in the algorithms. This will remove the effects of movements outside the cages. The algorithm can also be run starting at the point when the laboratory technician is already finished setting up the experiment.

It would be good if the algorithm can take the changes mentioned in (b) into account.

3. *cast shadows*: shadows can be formed from background objects (e.g. mouse shelter) or by the moving foreground objects themselves. This happens in laboratory situations. There are some algorithms which take this into account.
4. *bootstrapping*: empty (free of moving objects) background image(s) at the beginning of the sequence could be absent. There are laboratory experiments which start with empty background, but there are also some which start with the animal already in the cage. There are many algorithms which can estimate the background from obstructed parts of the video, as long as each pixel reveals the background for at least a short interval, or the foreground object does not remain stationary longer than the interval where the background is revealed.
5. *camouflage*: the moving objects to be detected could have chromatic features similar to those of the background model. In many laboratory situations, care is taken to clearly distinguish the background from the animal, like use a white straw bedding with a black mouse. There are some algorithms that use features other than color to model the background.
6. *structural changes*: background objects can be added or removed from the background, and the background itself can change in geometry and appearance. Examples are: car parked or moved out of the background, or a person stays stationary for an extended period. Because laboratory experiments are controlled, there are hardly structural changes occurring. But the animal can remain stationary for some time, which can be mistaken as part of the background.

It should be added here that because of the different changes that happen in the background, a completely stationary background may not exist. There is not one empty background image that could be considered as the only “correct” background, unless the reference background is defined as the image before the animal was introduced in the cage. Therefore, the extracted background is already considered acceptable when the animal (foreground object) is not present in the image.

Following are requirements of a good background extraction algorithm in laboratory situations:

1. The algorithm should perform fast. This is especially true if background extraction is just in the initial part of the tracking process.
2. The algorithm should output an empty background, that is, a background without the foreground object or animal.
3. The algorithm should work for different type of animals: fast or slow moving, small or large, one or more animals, one colored or multi-colored animals.
4. The algorithm should work for different types of setups: one arena or multiple arena setup, open field, Y maze, Plus maze, etc.

To summarize, the above list looks at the speed, quality (obtains empty background), and applicability (can be applied to different situations in the laboratory) of the algorithms.

Objectives

The objectives and research questions (RQ) of the study are:

Objective: Find a suitable background extraction algorithm that can be used in laboratory situations.

General research question: Which is the best algorithm?

Subquestions:

RQ1: Which algorithms can be tested, what are their characteristics and how good are they expected to produce an empty background?

RQ2: Which quantitative and reproducible measures can be used as criteria to evaluate the performance of the different methods?

RQ3: How can the evaluation be performed to get the best algorithm?

RQ4: How does the algorithm perform in the different laboratory situations?

Scope of the research

The study was concerned with the extracted background, as the output of the background modeling part of a background subtraction method. Detection and tracking of the animals themselves are not the output of the study.

The emphasis of the study is on acquiring algorithms and choosing the best one to extract background and to apply it. The applicability of the algorithm to different situations in the laboratory is much desired.

Only the algorithms that were available with source code were implemented and tested.

The study focused on the laboratory situation. The algorithm may be applied to outdoor situations but this was not considered in the study.

The study was limited to videos taken from a single static camera, and not from multiple or moving cameras. The videos to be used should be of good quality.

The report is divided into five sections. The next section presents a literature review on background subtraction algorithms, since background extraction is part of it. It also contains methods used in the literature to evaluate performance of algorithms. The methodology is described in Section III, and the results are presented and discussed in Section IV. The last section is on the conclusions and recommendations.

II. Literature Review

As has been explained in Section I, background subtraction involves background modeling where the background is calculated and updated. The background modeling process consists of model representation, model initialization and model adaptation (Cristani, et al., 2003). The first refers to the kind of model used to represent the background, the second deals with initialization of the model, and the third describes the mechanism used to adapt the model to changes in the background.

A. Background representation

The simplest background model is a known background. This can occur when the empty background is available at the start of the video. If this is not available then the background has to be modeled.

Many background modeling techniques have been developed and presented in surveys (Cheung, et al. 2004, Elhabian, et.al. 2008, Piccardi, 2004, Radke, et.al. 2005). These methods can be classified according to the complexity or nature of the method as in the following categories (Bouwman, et.al. 2008, El Baf, et al. 2008a):

1. Basic Background Modeling,
2. Statistical Background Modeling (parametric/non parametric, uni-modal/multi-modal)
3. Fuzzy Background Modeling
4. Background estimation

The methods can also be classified according to the features they use, which can be based on:

1. feature type (color, edge, stereo, motion and texture)
2. feature size or data-abstraction level (pixel, block or cluster or region, or frame).

A.1. Type of method

1. Basic Background Modeling – These methods model the background at each pixel location based on the pixels history (Piccardi, 2004). The methods are quite easy to employ. Some are statistical in type (e.g. median or average), but they are put in this category for their simplicity. The method can use a learning rate and selectivity parameter (e.g. Running Average). This includes the following methods (Elhabian, et.al., 2008):

Frame differencing – uses each previous video frame (time $t-1$) as the background model for the current frame (time t). The technique is sensitive to noise and changes in illumination.

Average filtering – takes the average of the images over time. All information from the background and foreground is used. The method is not robust with moving objects,

especially if they move slowly. It does not handle backgrounds which are bi-modal, and recovers slowly.

Median filtering – takes the median at each pixel location as the background. It assumes that in more than half of the frames in the buffer, the pixel shows the background.

Both average and median filtering can perform fast, but are memory consuming (Piccardi, 2004).

2. Statistical Background Modeling

- a. Uni-modal vs. multi-modal: this is employed when the intensity, color or other features of the pixel are assumed to follow a single uni-modal or multi-modal distribution. Uni-modal distributions implicitly assume a static background. Multi-modal distributions are used for modeling non-static or moving backgrounds.
- b. Parametric vs. non-parametric: parametric models makes assumptions on the data and uses parameters to describe the model. Non-parametric models use more flexible representation of the probability distribution at each pixel, without any assumptions on the underlying distribution. It can rely heavily on the data and can be computationally costly.

The following methods (Elgammal, 2011) belong to this group:

Single Gaussian Model – uni-modal and parametric. This model assumes that the intensity observed at each pixel is a random variable with a Gaussian distribution, $N(\mu, \sigma^2)$, with the mean and variance estimated from history of pixel observations. The method determines whether a new observation at each pixel comes from the estimated background distribution. Assuming a uniform foreground distribution, the classification rule marks a pixel as foreground if:

$$\|x_t - \hat{\mu}\| > k\hat{\sigma},$$

where $\hat{\mu}$ and $\hat{\sigma}$ are estimates of the mean and standard deviation, and k is a threshold. This model reduces to subtracting a background image from each new frame and checking if the difference is bigger than the threshold. Here the background image is the mean of the history of background images. The model can be adapted to slow changes in the background.

Mixed Gaussian Model (MoG) – multi-modal and parametric. Here the pixel intensity is modeled as a weighted mixture of more than one Gaussian distributions. This can represent the colors or different parts of the background. Stauffer and Grimson (1999) used a number of Gaussians from 3 to 5, and weighted the mixture using the frequency by which the background is explained by each of the Gaussians. The probability that a pixel has intensity x_t at time t is estimated as:

$$\Pr(x_t) = \sum_{i=1}^K w_{i,t} G(x_t, \mu_{i,t}, \Sigma_{i,t}),$$

where $w_{i,t}$, $\mu_{i,t}$, and $\Sigma_{i,t} = \sigma_{i,t} \mathbf{I}$ refer to the weight, mean and covariance for the Gaussian mixture component i at time t , respectively. K-means approximation is used to

update the parameters recursively. Each new pixel value is compared to existing K Gaussians and when there is a match, the weight, mean and variance for that distribution are updated.

The MoG model performed well in indoor and outdoor situations.

Kernel Density Estimation (KDE, Elhabian, et.al. 2008) – multi-modal and nonparametric. Kernel density estimators are applied to situations where the scenes contain dynamic areas, like waving trees and rippling water. It is a more flexible representation of the probability distribution of the background at each pixel. Given a sample x_1, x_2, \dots, x_N of intensity values for a pixel, the intensity $\hat{p}(x)$ of the density can be estimated using:

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^N K_{\sigma}(x - x_i),$$

where K_{σ} is a kernel function (also called “window” function), and σ is the bandwidth (scale), such that $K_{\sigma}(t) = \frac{1}{\sigma} K(\frac{t}{\sigma})$, $K(t) \geq 0$ and $\int K(t) dt = 1$. With sufficient samples, KDE estimators converge to any density function asymptotically. It is a general approach in the sense that it does not assume any specific density function. It can be extended to use color or other high dimensional features. Using the probability estimate $\hat{p}(x)$, the pixel is considered part of the foreground if the estimate is less than a global threshold value over all the image. The global threshold can be adjusted to achieve a desired percentage of false positives. The N samples can be determined using a sliding window over time.

3. Fuzzy Background Modeling – applies Fuzzy theory in background modeling. This can involve using a saturating linear function instead of a hard limiter function. The result will be a real value in the [0,1] range. For example, in Fuzzy running average, the learning rate is not a fixed overall value, but is based on the current value of fuzzy background subtraction, which can be computed for each pixel (Sigari, et al., 2008). Fuzzy Gaussian, uses a fuzzy classification rule and employs fuzzy on-line cumulative averages for selective update of the mean and the covariance matrix. The selective update of the background provides better segmentation of the foreground objects than simple Gaussian (Bender, 2013). The Fuzzy Mixture of Gaussians Model (El Baf, et al., 2008b) improves on the Mixture of Gaussians model for critical situations like presence of camera jitter, waving trees and rippling water.

Another method used fuzzy integral to fuse the texture and color features or edge and intensity for background subtraction (El Baf, et al., 2008a, Zhang and Xu, 2006). Maddalena and Petrosino (2010) proposed a spatial coherence variant to self-organization through artificial neural networks method and used a fuzzy model to deal with problems of using crisp settings.

4. Background Estimation – the background model is estimated using a filter. Any pixel that significantly deviates from its predicted value is considered foreground. Examples are Wiener filter, Kalman filter, and Tchebychev filter (El Baf, et al., 2008a).

There are other methods that are extensions of the methods mentioned above. For example, instead of using one value for learning rate or fixing the number of components per pixel as in some methods above, different learning rates or number of components are used (Bouwman, et

al., 2008, Zivkovic and van der Heijden, 2006)). Some methods utilize different equations at different phases (KaewTraKulPong and Bowden, 2001).

A.2. Classification based on features

1. Feature type: some of the features used in calculating backgrounds are intensity, color, and texture. Some methods also use edges or motion in the algorithms.

If the camera is fixed and the background stays constant, it can be modeled using a single static image. The measurement used is the intensity for gray level images, or color components for color images. If the background is not constant, then the mean intensity and variance of a pixel can be modeled to adapt to the variation in the background. If the background contains motion, then more tolerant models are required. One approach predicts the motion pattern of each pixel, using optic flow. However, this can fail when the motion fields of the foreground and background pixels are not different. Another type of background model uses a linear predictor of the intensity of a pixel using history of intensity values in that pixel. This can account for periodic variations. Hidden Markov Models can represent different states of a pixel, e.g. background, foreground, shadow, and sudden changes in illumination. Group of clusters which are ordered according to the likelihood that they model the background can also be used.

2. Feature size or data-abstraction level

Following are the methods developed based on feature size:

- a. Pixel –level processing – is low level processing of each pixel independently. It classifies the pixel as either foreground or background, and manages adaptation to changes in the background. This level can have problems with local or global sudden illumination changes.
- b. Block or region – level processing – here the image is divided into blocks and special features of the block are used in the calculation (Zhang and Xu, 2006). This is considered a higher level representation; modeling also inter-pixel relationships. For example, the spatial motion of the foreground is detected by segmenting foreground patterns and intersecting successive segmentations. Reddy et al. (2011) based their method on combined frequency response of blocks and their neighborhoods.
- c. Frame – level processing – looks for changes in large parts of the image and swaps in more expressive background models.
- d. Combination of pixel and region – integrates pixel level information with region information obtained from spatial segmentation of the background. The spatial segmentation can be done using chromatic, spatial and temporal information (Cristani, et al., 2003) or optical flow of blocks between successive frames (Reddy, et al., 2011). The use of optical flow can make the processing computationally intensive.

B. Background initialization

Most background models use initial parameters which are derived from short sequences where no foreground objects are present. Sometimes this is difficult to achieve, and the model has to be “trained” using a sequence which contains foreground objects. Several assumptions are made to make this possible. Gutchess, et al. (2001) said that each pixel in the frame should reveal the background for at least a short time interval. Moreover, the background should be close to stationary or background motion may occur but should be small only. Wang and Suter (2005) added that a foreground object can appear stationary for a short time, but no longer than the interval where the background is revealed.

Some methods used for model initialization are:

1. Median Filtration – assumes that the background at each pixel is visible more than 50% of the time during the training sequence.
2. Stable intensity extraction – this uses the longest, most stable interval to represent the background. This has problems when foreground objects stay stationary for a long period of time.
3. Relative Constant Intensity Extraction – this is similar to stable intensity extraction, but overcomes the stationary foreground problem by considering the optical flow in the neighborhood around each pixel. This approach suffers from computational complexity and sensitivity to noise.
4. Background with a Mixture of Gaussians Distributions – this estimates the values that are used for the parameters of the mixture of Gaussians distributions. The estimates can be calculated offline using expectation maximization or use an online data-driven adaptive approach.

Other methods are Kernel Density Estimation, Hidden Markov Models, Codebook based and others. Many of the methods mentioned in the Introduction can be used at this stage.

C. Background adaptation

Background adaptation methods can be classified into either predictive or non-predictive. Predictive methods use time series and dynamic models estimate current input from past observations. The difference between the predicted and actual are considered measure of change. Non-predictive methods build a probabilistic representation of the observations at the current pixel.

Another classification used for background adaptation is based on the history of the frames that are used in updating the model:

1. Recursive: updates the background model based on each input frame. Input frames from distant past can have an effect on the current background. Some schemes use exponential weighing to discount the past. In any case, an error in the model can be carried out for a long time. Recursive techniques require less storage. Examples of this type of technique are: Approximated median filter, Kalman filter, Single Gaussian or Mixture of Gaussians, Clustering-Based, and Hidden Markov Models (Cheung and Kamath, 2003, , Elhabian et al., 2008).
2. Non-recursive: stores a buffer of a certain number of previous frames and estimates the background based on the temporal variation of each pixel within the buffer. It is highly adaptive because they do not depend on history beyond those included in the buffer. The memory requirements can be significant if a large buffer is used, but there are ways to partially

alleviate this (e.g. lower the frame rate). Examples of non-recursive methods are Frame Differencing, Average/Median/Maximum-minimum filter, and Linear predictive filter (Cheung and Kamath, 2003, Elhabian et al., 2008).

There are other classification types mentioned in Elhabian et al., 2008. One is about selective or blind update. Selective update adds the new frame only if it is classified as a background sample. This could enhance detection of targets, but can lead to a deadlock situation when incorrect detection decision persists. Blind update adds new samples to the model. It can lead to bad detection of targets as they falsely become part of the model.

There are also short-term and long-term models. The first is very recent model, adapts quickly to allow sensitive detection and updates each sample using selective-update mechanism. Long-term model captures a more stable representation and adapts slowly. The update mechanism used is blind update.

Other considerations

Piccardi, (2004) discussed several methods and summarized them according to speed and memory requirements:

1. Speed

- Fast: Average, Median, Running average
- Intermediate: Mixture of Gaussians, KDE, Eigenbackgrounds, SKDA, Optimized mean-shift
- Slow: Standard mean-shift

2. Memory requirements

- High: Average, Median, KDE, Mean-shift
- Intermediate: Mixture of Gaussians, Eigenbackgrounds, SKDA
- Low: Running average

Background extraction algorithms

The internet was searched for background extraction algorithms. The BGSLibrary (Sobral, 2012) was found to contain 24 background subtraction algorithms which contained components that model the background. Radke et al. (2005) mentioned algorithms which were available from Andra and Al-Kofahi (2004). The algorithms were examined, but were considered not different from the ones already available in BGSLibrary.

The algorithms in BGSLibrary came with a range of characteristics, like types, modality, data abstraction level, feature size, etc., (see Appendix I). There were 8 Basic methods, 9 Statistical methods, and 7 Fuzzy methods. From the Statistical and Fuzzy methods, 12 were multi-modal and 4 were uni-modal. The multi-modal methods were especially developed to consider changes happening in the background. Also from the Statistical and Fuzzy methods, 6 were non-parametric and 10 were parametric. Gaussian distribution (single or mixture) was used by the parametric types. Of the 24 methods, 16 were recursive and 8 were non-recursive. Non-recursive methods can be

memory consuming when a large buffer is used. Most of the methods used pixel as data abstraction level or feature size, and color as feature type. Most methods are non-predictive. The methods have been tested in different situations by their authors.

In general, basic methods like FrameDifferencing, mean-based and median-based methods are simple to calculate and pretty fast. But they use a global threshold which does not change in time. Other algorithms deal with changes happening in the background (see Challenges to background modeling in the Introduction). The algorithms that are based on Mixture Of Gaussian model (LBMixtureOfGaussians, MixtureOfGaussianV1, MixtureOfGaussianV2, DPGrimsonGMM, DPZivkovicAGMM, T2FGMM_UM, and MultiLayer_Learn) consider changes due to light, moving background, structural changes, and cast shadows. The Self Organizing Map (SOM) models (LBAadaptiveSOM, LBFuzzyAdaptiveSOM) consider the changes due to light, moving background, camoufladge, bootstrapping, and cast shadows. DPPratiMediod considers shadow information. DPWrenGA deals with bootstrapping. FuzzyChoqueIntegral considers cast shadow and light changes, whereas FuzzySugenolIntegral considers moving background and light changes.

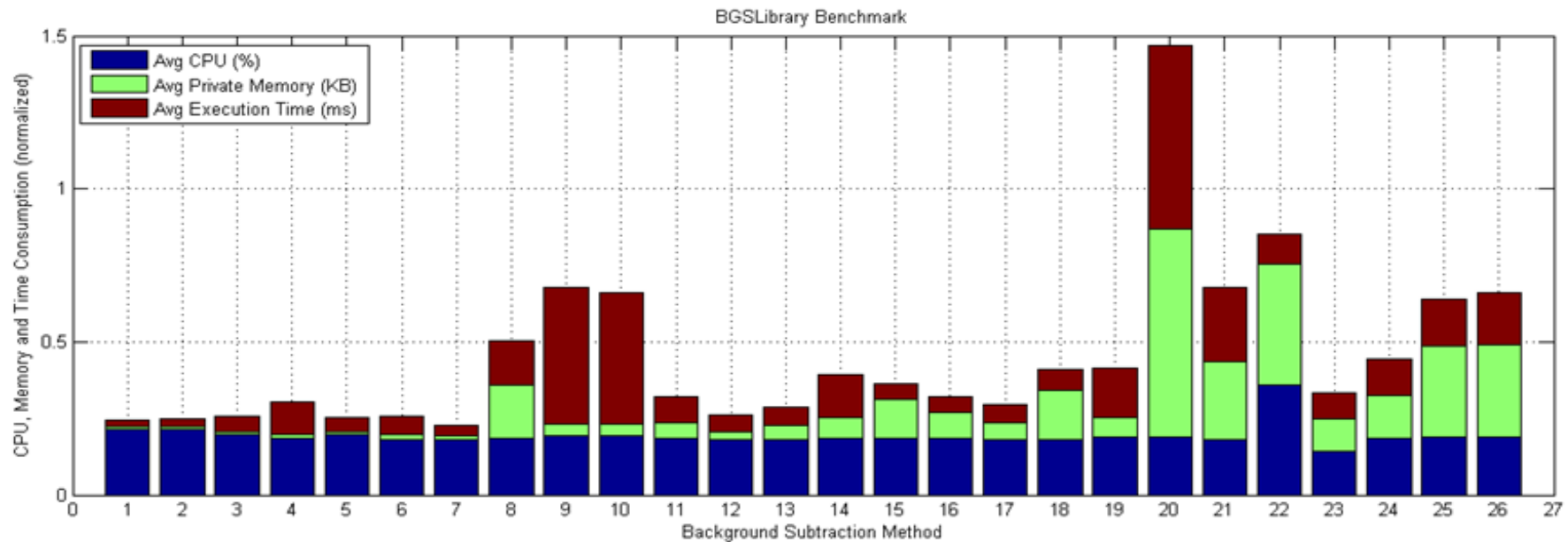
In 2013, Sobral compared the performances of the 24 algorithms, as well as newly added ones. His results are given in Figure 2. Except for two algorithms, the average CPU use of most algorithms are about the same. Most basic modeling algorithms (algorithms 1-7) use low average private memory and low average execution time. The Fuzzy methods 9 and 10, as well as the multi-layer method 20, use high average execution time. Methods 8, 20-22, and 25-26 use relatively longer average memory.

The descriptions, classifications and features of the different methods are important to consider in choosing the method for background extraction in laboratory situations. Specifically, methods that can provide the empty background fast and with not much complexity and memory requirements are preferred.

Evaluation Method

Since part of the objective is to find a quantitative and reproducible measure to use as criteria to evaluate the performance of the different algorithms, literature was searched on available methods to use. Elhabian, et. al. (2008) discussed evaluation of algorithms that are used in separating objects and their background (also called video segmentation). Since this study is on extracting background, which entails separating the background from the foreground, it was thought that the methods they used could be helpful. They gave two alternatives: standalone evaluation, when the reference segmentation is not available (so-called ground truth) and relative evaluation, when the reference segmentation is available for comparison. Standalone evaluation provides qualitative information for the ranking of algorithms, and work mainly with available a priori information on the expected properties of objects and their difference to neighboring objects. Relative evaluation is expected to provide more reliable results. Since the objective of the study is to use quantitative measures, relative evaluation methods were examined.

There are three approaches in relative evaluation: pixel-based, template-based and object-based. Pixel based includes all active pixels in a given image. It is a binary detection problem based on the ground truth. Example measures are misdetection rate, false alarm rate, receiver operating characteristics (ROC) and perturbation detection rate characteristics. Template-based and object-based methods were deemed not relevant since the interest of the study is the background and not object or foreground detection.



Legend:

- | | | | | |
|------------------------------|-------------------------|---------------------------|--------------------------------|-----------------------|
| 1 StaticFrameDifferenceBGS | 7 DPAdaptiveMedianBGS | 13 LBSimpleGaussian | 19 T2FGMM | 25 LBAadaptiveSOM |
| 2 FrameDifferenceBGS | 8 DPPratiMediodBGS | 14 DPGrinsonGMMBGS | 20 MultiLayerBGS | 26 LBFuzzyAdaptiveSOM |
| 3 WeightedMovingMeanBGS | 9 FuzzySugenolIntegral | 15 MixtureOfGaussianV1BGS | 21 PixelBasedAdaptiveSegmenter | |
| 4 WeightedMovingVarianceBGS | 10 FuzzyChoquetIntegral | 16 MixtureOfGaussianV2BGS | 22 GMG | |
| 5 AdaptiveBackgroundLearning | 11 LBFuzzyGaussian | 17 DPZivkovicAGMMBGS | 23 VuMeter | |
| 6 DPMBGS | 12 DPWrenGABGS | 18 LBMixtureOfGaussians | 24 DPEigenbackgroundBGS | |

Computer Configuration:

Machine: Intel Core i5-2410M CPU 2.3Ghz
RAM: 4,00GB DDR3
OS: Windows 7 x64 Home Premium SP1

Input:

BMC dataset - EvaluationPhase - Street - 212
Image resolution: 640x480 - 24bits
Number of images: 1499

Figure 2. CPU, Memory and Time Consumption of Background Subtraction algorithms in BGSLibrary (from Sobral, 2013).

From the literature, ground truth can be generated using synthetic data, like ellipse fitting, edge detection, corner detection and optic flow. Another method is manual annotation, e.g. to mark edge and no-edge pixels. The third approach relies on evaluating the output of the algorithms by a human panel (e.g. all vote, majority rule, set union or consensus markup). The first two methods are more appropriate to detecting foreground objects than extracting the background. The last methods is not objective and is therefore not very satisfactory. Another method of generating the ground truth has to be devised.

In pixel-based evaluation methods, the following quantities were used in comparing the ground truth to a candidate binary foreground map:

True positives (TP) : number of foreground pixels correctly detected

False positives (FP): number of background pixels incorrectly detected as foreground (“false alarms”)

True negatives (TN): number of background pixels correctly detected

False negatives (FN): number of foreground pixels incorrectly detected as background (“misses”)

Since the study is on background extraction, the above quantities has to be translated into “background” terms as in the following:

True positives (TP) : number of background pixels correctly detected

False positives (FP): number of foreground pixels incorrectly detected as background (“false alarms”)

True negatives (TN): number of foreground pixels correctly detected

False negatives (FN): number of background pixels incorrectly detected as foreground (“misses”).

For a given “empty” background which is extracted by an algorithm, the whole output is then supposed to be the background. From the 4 quantities above, TN and FN are not relevant.

Cohen and Medioni (1999) proposed metrics for moving object detection evaluation. The *False Alarm Rate (FAR)* shown below looks appropriate for the study:

$$FAR = \frac{FP}{TP + FP}$$

Maddalena and Petrosino (2008, 2010) used a similar metric which they called Precision or positive prediction. According to them, Precision gives the percentage of detected true-positive pixels as compared to the total number of pixels detected by the method. The metric is shown below:

$$\text{Precision} = \frac{tp}{tp + fp}$$

As can be seen in the formula’s one is the opposite of the other. Whereas FAR emphasizes on incorrectly detected pixels, Precision emphasizes on the correctly detected pixels. Another way of saying this is that FAR measures how different the detected pixels are from the ground truth, whereas Precision measures how similar the detected pixels are from the ground truth.

III. Methodology

The study is concerned with finding an algorithm that can extract the background from videos taken in the laboratory and applying the algorithm in different situations. From the requirements stated in the Introduction, a good algorithm is fast, has good quality (produces an empty background) and applicability (can be applied to different types of animals and setups). The methodology that was designed to meet the requirements, achieve the objectives and answer the research questions are given below:

Find a suitable background extraction algorithm – Which is the best algorithm?

1. **Which algorithms** can be tested, what are their characteristics and how good are they expected to produce an empty background?

A total of 24 algorithms from BGSLibrary (Sobral, 2012) were selected for the study. The program codes for BGSLibrary were downloaded from Sobral (2012) and adapted to run in Visual Studio 2010 with C++ as the programming language. The OpenCV library (OpenCV, 2013) was used in the program.

The algorithms were analyzed and expectations on how they would perform in producing empty backgrounds were made.

2. **Which quantitative and reproducible measures** can be used as criteria to evaluate the performance of the different methods?

The quantitative and reproducible measures to use for evaluating the algorithms should answer the requirements set at the beginning of this study (Introduction). The requirements summarize to speed, quality and applicability. At this stage of the study, only the first two were be addressed. This is because for applicability, the algorithms would have to be tested in different situations using different videos, and that would take a lot of effort. The decision was then to first find the best algorithm using the speed and quality criteria, and then test the chosen algorithm if it can perform well in different situations (Step 4).

Since the criteria will be used to compare different algorithms, they were chosen so that they are independent of the algorithms themselves and independent of each other. Speed was measured by taking the time it took to run the same number of frames. For the quality measures, a metric that is similar to False Alarm Rate (FAR, see Literature Review) was used. It was measured by determining how different each extracted background was from a ground truth, the difference being considered as equivalent to the “incorrectly detected” pixels. The minimum of the difference values over all the frames, as well as the frequency by which the minimum occurred were considered as measures of quality. The frequency indicates how often the extracted background was close to the ground truth. The best algorithm would ideally have the lowest difference and the most frequent occurrence of the minimum value. Note that in contrast to FAR, the proportion of incorrect detection is not computed. This is to preserve the precision of

the difference values. Moreover, since all algorithms use the same video, the denominator in FAR will all be the same value.

Two representative algorithms were selected as examples to see how good the criteria were able to differentiate the performance of the algorithms. They were run with 7 different videos. The following were done to study the performances:

- a. The disappearance and appearance of animals, which determines the rate of obtaining the empty background were examined.
- b. The speed it took to run the same number of frames were calculated.
- c. For one of the videos, the values of the quality measures were computed for extracted backgrounds using the two algorithms.

At the end of this exercise, the speed and quality measures were found to satisfactorily differentiate the two algorithms. The identified measures were used as criteria to evaluate the 24 algorithms in the following step.

3. **How can the evaluation be performed** to get the best algorithm?

The following steps were done in the evaluation:

- a. All the algorithms were run using the same video to calculate the speed and quality measures. Two videos were used, which differed in complexity and color of the animal. More videos could have been used for this step, but since further analyses will be done on the results, it was decided to better limit the number of videos and spend more time in analyzing the results.

The images in the videos were masked so that only the arena areas were used. The reference images were taken at the start of the videos before the animal was put in the case. The algorithms were run starting at a point in the video when the animal is already present. The algorithm calculated/extracted backgrounds were taken and compared with the reference backgrounds. The speed and measures of quality were calculated. The frames where the measures of quality occurred were checked if the extracted background were without traces of the animal. The calculated values were inspected to see if they have relation with the characteristics of the algorithm, such as type of algorithm and modality.

- b. Several multiple criteria evaluation (MCE, Heywood, et al., 2006) were performed using the different measures (criteria). MCE analysis is commonly used when choosing the best option (e.g. the best algorithm) given different considerations (criteria). Three MCE analyses were made corresponding to three scenarios. These scenarios represent different preferences of users of the outputs. They were:

Speed = Quality: equal weights were given to Speed (50%) and Quality (50%).

Speed > Quality: more weight was given to Speed (60%) than Quality (40%).

Speed < Quality: less weight was given to Speed (40%) than Quality (60%).

For the MCE, the values for the different criteria were standardized before applying the weights and obtaining the scores. The quality weights were distributed to the different measures of quality. The algorithm with the highest score was taken as the best in the list.

The MCE analyses were extended by doing a sensitivity analysis on the frequency measure. The results were obtained for the cases when the frequency were calculated for minimum difference plus 10 pixels, plus 25 pixels, plus 50 pixels and plus 100 pixels. The results would indicate which algorithms have frequently low values of the differences.

4. **How does the algorithm perform** in the different laboratory situations?

The best algorithm was tested using 37 videos representing a wide range of situations in the laboratory. This includes situations with different animal size and color, constant and non-constant background (water, beddings), arena's with reflections and poor lighting, arena's containing novel objects, low contrast animals, shadows, long tails, etc. There were experiments with single and multiple arenas. The set also includes movies for which the algorithm is known to fail (animal hardly move). It was expected that the videos were of good quality. Each pixel in the video should reveal the background for at least a short interval.

Before the images in the videos were subjected to the algorithm, they were first masked so that only the part inside the arenas were left. This was done to increase speed (area outside arena were not important and need not be calculated). For multiple arenas, the algorithm was applied to all arenas at the same time.

For the runs, the values of the parameters in the algorithm were initially set equal to the values used in the BGSLibrary. The program was made such that it is possible for the user to change the parameter values to better adapt to the situation. For example, for a fast animal, a higher learning rate can be used.

The runs started at the beginning of the video. Once the run proceeds, a feedback on how the background changes was shown. The user could stop the run at any time and use the resulting background extracted.

The extracted images from the 37 videos were analyzed to see in which cases the algorithm performed well and in which cases it did not perform well.

A summary of the methodology is shown in Figure 3.

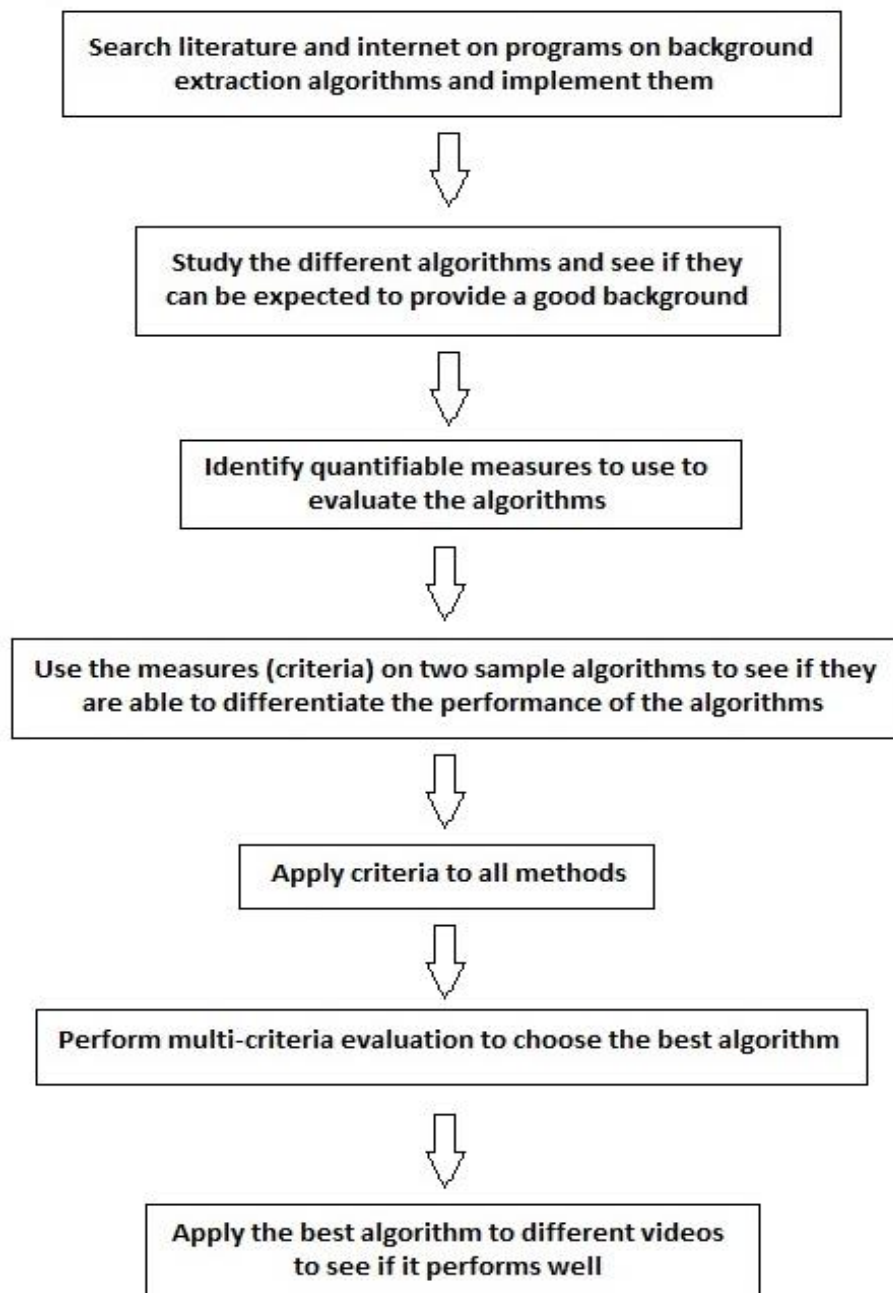


Figure 3. Steps undertaken in the methodology

IV. Results and Discussion

Find a suitable background extraction algorithm – Which is the best algorithm?

1. Which algorithms can be tested, what are their characteristics and how good are they expected to produce an empty background?

The BGSLibrary (Sobral, 2012) contained 24 background subtraction algorithms or methods. These algorithms have components that extract the background. The programs were made to run in Microsoft Visual Studio 2010 with C++ as the programming language. Default values of the parameters in the algorithms were used.

The algorithms came with a range of characteristics, like types, modality, data abstraction level, feature size, etc., (see Appendix I). After examining the descriptions, the literature, the source codes and results of sample runs, the following are the expectations on each of the algorithms:

AdaptiveBackgroundLearning – it is not possible to get a background that is completely “clean” or empty. There is always a trace of the animal left, especially if the animal is visible at all times, when the color of the animal is different from the background and when the alpha value is small.

DPAdaptiveMedian – the algorithm works well for simple backgrounds, but may have problems extracting empty backgrounds when there are significant changes occurring in the background and if the animal stays stationary more than half of the time.

DPEigenbackground – the background was updated in the program; runs with the algorithm produced backgrounds which were dark, which make the background unusable.

DPGrimsomGMM – the algorithm, as created by the authors of the paper (Stauffer and Grimson, 1999), had been the basis of other mixture of Gaussian algorithms. Runs with the algorithm showed the background continuously being adapted with appearance and disappearance of the animal. Empty background can be obtained with the algorithm.

DPMean - the algorithm is simple and quite fast, but runs of the algorithm always showed the animal in the background.

DPPratiMediod - it is possible to get an empty background with this algorithm, especially if the animal does not stay stationary for more than half of the time.

DPWrenGA - without adjusting the parameters for the animal in the laboratory, runs of the algorithm outputted backgrounds which were sometimes free of animal.

DPZivkovicAGMM – the background is continuously updated; it is possible to get a background free of animal.

FrameDifference – the algorithm is simple and fast, but it is impossible to get an empty background with the algorithm if the animal is always in view.

FuzzyChoqueIntegral – the backgrounds were updated during the learning period in the algorithm, but the outputted backgrounds were completely dark and therefore unusable.

FuzzySugenIntegral – the backgrounds were updated during the learning period in the algorithm, but the outputted backgrounds were completely dark and therefore unusable

LBAptiveSOM – runs with the algorithm showed the background being updated, like traces of the animal disappearing. It is possible to get an empty background with the algorithm.

LBFuzzyAptiveSOM - runs with the algorithm showed the background being updated, like traces of the animal disappearing. It is possible to get an empty background with the algorithm.

LBFuzzyGaussian – this is supposed to be an improvement of LBMixtureOfGaussian. It is possible to get an empty background with this algorithm.

LBMixtureOfGaussian – this is based on the mixture of Gaussians distribution like DPGrinsonGMM; the changes in the background are considered, and it is possible to get an empty background with this algorithm.

LBSimpleGaussian – it is possible to get an empty background with this algorithm however, being unimodal it is limited in dealing changes in the background.

MixtureOfGaussianV1 – the algorithm did not output any background. There was an assertion error in using the OpenCV function. Other people reported similar problem in using the function in the internet.

MixtureOfGaussianV2 – the algorithm did not output any background. There was an assertion error in using the OpenCV function. Other people reported similar problem in using the function in the internet.

MultiLayer – the learn phase of this algorithm could produce a background free of animal, the background produced with one video appeared layered.

StaticFrameDifference – if the first frame in the video has an animal, then the background will always contain an animal, otherwise, the background is empty.

T2FGMM_UM – supposed to be an improvement to the ordinary mixture of Gaussian algorithm; runs of the algorithm showed the background being updated; could produce an empty background

T2FGMM_UV – runs of the algorithm showed the background being updated; could produce an empty background

WeightedMovingMean – runs of the model showed the background always filled with traces of the animal.

WeightedMovingVariance – runs of the model showed the background was not produced at all.

From the above list, one can say that many of the algorithms fulfill the characteristics described in Section IV A, like good speed and quality. It is possible to obtain empty backgrounds with many of them. Exceptions are the following:

- Some algorithms produce backgrounds that contain the animal or traces of the animal. This includes the algorithms: AdaptiveBackgroundLearning, DPMean, FrameDifference, and StaticFrameDifference.
- Other algorithms like DPEigenbackground, FuzzyChoquetIntegral and FuzzySugenIntegral produced backgrounds which were dark or all black.
- The algorithms MixtureOfGaussiansV1 and MixtureOfGaussiansV2 resulted in run errors.
- The WeightedMovingVariance algorithm did not produce any background.

Looking at the speed indications in Figure 2, the following groups could be made according to relative speed of the algorithms:

Slow: MultiLayer, FuzzySugenIntegral, FuzzyChoquetIntegral, LBFuzzyAdaptiveSOM, T2FGMM, LBAadaptiveSOM, DPPratiMediod and DPGrimsomGMM
Fast: StaticFrameDifference, FrameDifference, DPAdaptiveMedian
Intermediate: all the rest.

According to Figure 2, the following algorithms use lots of private memory: MultiLayer_Learn, LBFuzzyAdaptiveSOM, LBAadaptiveSOM, DPPratiMediod, LBMixtureOfGaussians, DPEigenbackground and MixtureOfGaussiansV1. Most computers nowadays are equipped with considerable memory, so the requirements of some of these algorithms may not be a problem.

2. Which quantitative and reproducible measures can be used as criteria to evaluate the performance of the different methods?

The following measures were identified as possible criteria for evaluating the algorithms: Speed, minimum Difference, and the frequency of occurrence of Difference measure. Speed measured the amount of time it took to run the same number of frames. The other criteria measured the quality of the extracted background. They measured how different the extracted background was to the ground truth or reference background. The ground truth was the background without any animal, taken at the start of the video before the animal was put in the cage. The background extraction algorithm was started at the point in the video when the animal was already in the cage. The background was calculated or “extracted” for each succeeding frame in the video. Each extracted background was then compared to the reference image by taking the difference between the extracted and reference backgrounds. The lowest (minimum) of these values was taken as measure of the quality of the algorithm. The frequency by which the minimum occurred was also added as an additional measure of quality to indicate how often an algorithm can output a background that is not different from the reference background.

The values of the Difference metric was calculated using OpenCV's functions:

```
cv::Mat img_diff;  
cv::absdiff(img_start, img_output, img_diff);  
int iDiff = cv::countNonZero(img_diff);
```

where:

img_start: is the reference image, taken from the start of the video when the animal was not in the cage yet.

Img_output: contains the extracted background image of an algorithm for a given input frame

img_diff: contains the difference values between img_start and img_output

cv::absdiff : is an OpenCV method for calculating the absolute difference of corresponding pixels of two images. If corresponding pixels are the same, then the value 0.

cv::countNonZero: is an OpenCV method that counts the number of non-zero pixels, i.e., the number of pixels were difference between the images were observed.

In another calculation, the difference values were first “thresholded” before the number of non-zero values were counted. This was done using the following:

```
cv::Mat img_thrDiff;
cv::threshold(img_diff, img_thrDiff, 30, 255, cv::THRESH_BINARY);
```

where:

img_thrDiff: contains the thresholded difference values

cv::threshold: is an OpenCV method that applies threshold operation for each pixel in the image.

The threshold function cv::threshold is used to filter (e.g. remove noise) and segment values of the pixels. With the CV_THRESH_BINARY parameter in the threshold function, a binary image can be obtained where a cut off value for pixels between 0 and 255 can be made. For the study the cut off value was set to 30. This value was taken after testing with different values. Pixel values above this threshold value get the maximum value set (here it is 255); and all other pixels get the value 0. The number of non-zero values are then limited to values above the threshold. The pixel difference values under the threshold were considered to be noise, since it is impossible to compare 2 images which are exactly alike due to change in conditions over time.

The minimum “thresholded” difference and its frequency were used as candidate quality measures together with Speed.

Testing the identified criteria on two sample algorithms

The identified criteria were tested to see if they can differentiate the performance of the different algorithms. The 24 algorithms from BGSLibrary were shortly examined and two algorithms were chosen for the test. They were DPGrimsonGMM (GrimsonGMM) and LBMixtureOfGaussians (LBMOG).

- a. The algorithms were first examined by looking at how the animals “disappear” and “appear” in the backgrounds. The disappear and appear events can be seen while the algorithms update the backgrounds. For example, for both GrimsonGMM and LBMOG, when the animal moves a lot, the picture of the animal slowly disappears in the background. On the other hand, when the animal stays in the same location for a certain time, the background starts to show the animal again. The rates of appearance and disappearance indicate how sensitive the algorithms are to the movements of the animals. They affect the time the empty background can be extracted or maintained by the algorithm. Ideally, later appearance and earlier disappearance are better for empty background extraction.

The examination was done by running both algorithms using the same video at the same time. A sample interval of 1 (all frames considered) was used in most cases. Seven videos were used for the examination. They represent different situations which can be encountered in the laboratory. They were:

- green: “open field” arena containing, 2 mice
- BM1: multiple (6) arena, rectangular shape, one rat per arena (in 5 arenas)
- BS8: Y maze, slightly varying lighting conditions, 1 mouse
- BS9: Zero maze (backlight), darker part in 2 sides, 1 mouse
- BD15: arena (small aquarium), 1 small zebra fish
- BS2: rectangular arena, 2 background objects, 1 hooded rat
- BD4: rectangular arena, 2 mouse with lots of small background objects (droppings?)

The results are given in Appendix II. In the figures, the video frame when the algorithm was started are shown. In some videos, the algorithm started with an animal in the cage, whereas in others not. It can be seen that in general, appearance of the animal occurred earlier in LBMOG (rows 2, 3, 5, 7, and 9). The disappearance of the animal occurred earlier in GrimsonGMM in rows, 4, 6, and 8. However, in some cases, the disappearance of the animal also occurred earlier in LBMOG (rows 1, 10, 11, 12). The rates of appearance and disappearance affect the rate by which the empty background is extracted or maintained. It is affected by the learning rate parameter in the model. Generally, a higher learning rate should be used with a faster animal.

- b. The speed it takes to run the same number of frames were also examined. The two algorithms were run separately. The same 7 videos as above were used. The speed of running 500 frames using a sample interval of 1 were calculated. Table 1 and Figure 4 show the results

Table 1. Speed (msec) results for LBMOG and GrimsonGMM using 500 samples

Number	Video	LBMOG	GrimsonGMM
1	green	0.12	0.12
2	BM1	13.22	15.97
3	BS8	32.26	43.48
4	BS9	13.25	15.30
5	BD15	33.59	43.94
6	BS2	14.79	21.97
7	BD4	20.37	23.23

As can be seen in the table and figure, LBMOG is in general faster than GrimsonGMM for all seven videos. Only exception is the “green” video. This means that the earlier disappearance in GrimsonGMM that were observed in some videos in Appendix II, actually occurred later in time than in LBMOG. The results in the table is also consistent with what is shown in Figure 2. In the figure, LBMOG (Method 18) has a lower average execution time than GrimsonGMM (Method 14). This means that in terms of speed, LBMOG is better.

From the above results on different videos, it can be said that Speed can be used as one criteria to differentiate algorithms.

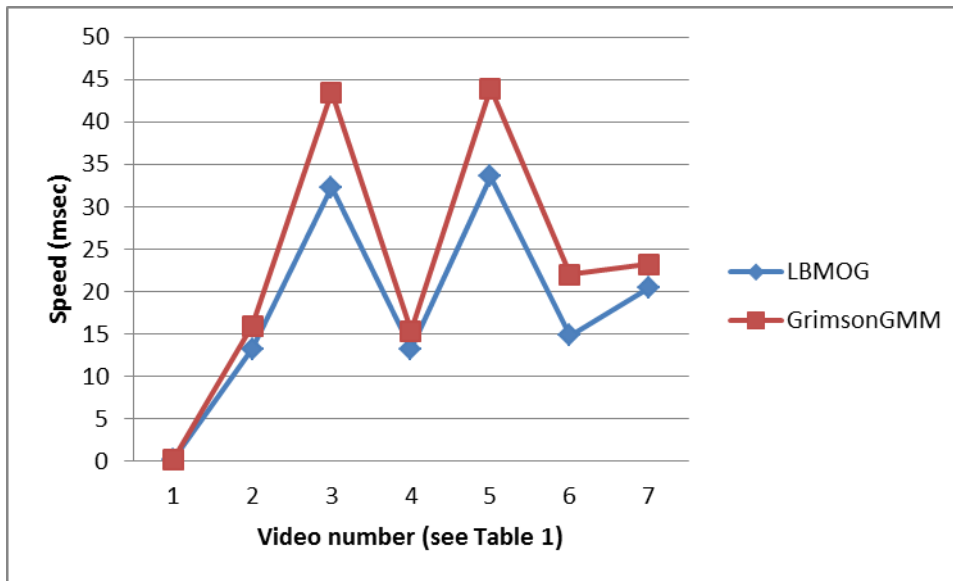


Figure 4. Speed results for LBMOG and GrimsonGMM using 500 samples

- c. The values of the quality measures were calculated after comparing the reference and “calculated” (extracted) backgrounds.

The BS2 video was used in the evaluation activity because it has an “empty background” at the start of the video. Representative frame at the start of the video was taken to serve as reference image. The LBMOG and GrimsonGMM algorithms were run from the time the animal was already present in the video using a sample interval of 1. The extracted background images were compared with the reference image by calculating the Difference and “thresholded” Difference values for each frame input. The minimum and maximum Difference values, as well as the frequencies of the minimum values, are given in Table 3 and shown Figures 5 to 8.

Table 3 shows that the minimum and maximum difference values in LBMOG are lower than the corresponding values in GrimsonGMM. A Difference minimum value of 0 was even derived after thresholding. The number of occurrences of the minimum Difference values show that there were 768 counts of the 0 “thresholded” Difference value in LBMOG. There were 2 counts of the minimum thresholded Difference value for GrimsonGMM. These results show that LBMOG is a good method.

Table 2. Minimum and maximum values for runs with BS2, sampling interval 1 (Number represents the frequency of the same value of minimum Difference).

Algorithm	Statistic	Difference	Difference Threshold
LBMOG	Minimum	56953	0
	Maximum	71919	3207
GrimsonGMM	Minimum	59649	1477
	Maximum	72337	4564
LBMOG	Number	1	768
GrimsonGMM	Number	1	2

Figures 5 and 6 show the trends in the Difference values using video BS2 for both LBMOG and GrimsonGMM. The scales in the figures were made similar. The figures show that the rates of increase and decrease in values are more pronounced in LBMOG than in GrimsonGMM. The Difference values in LBMOG are in general lower than in GrimsonGMM.

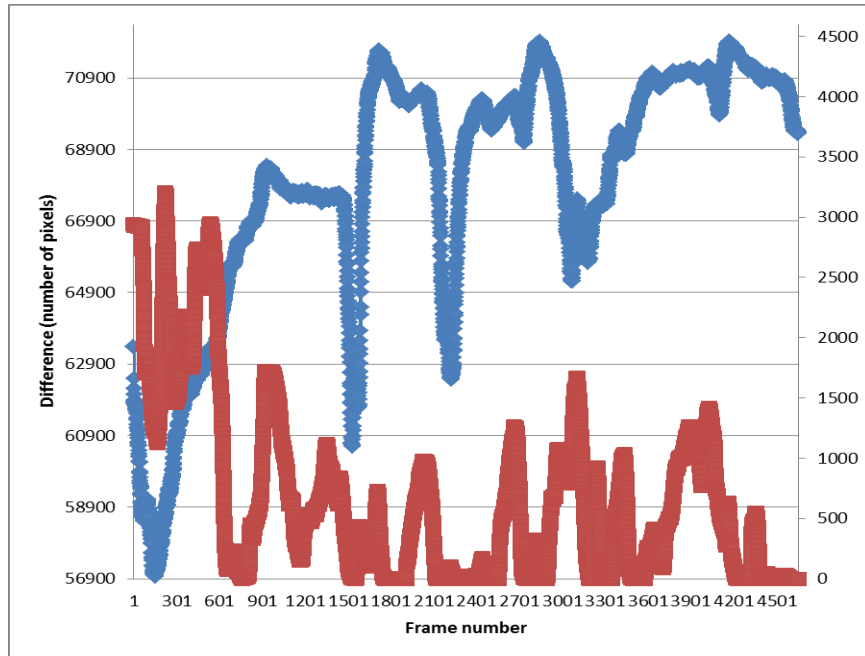


Figure 5. Difference values for LBMOG using video BS2, sampling interval 1, (normal values in blue, left axis; thresholded values in red, right axis).

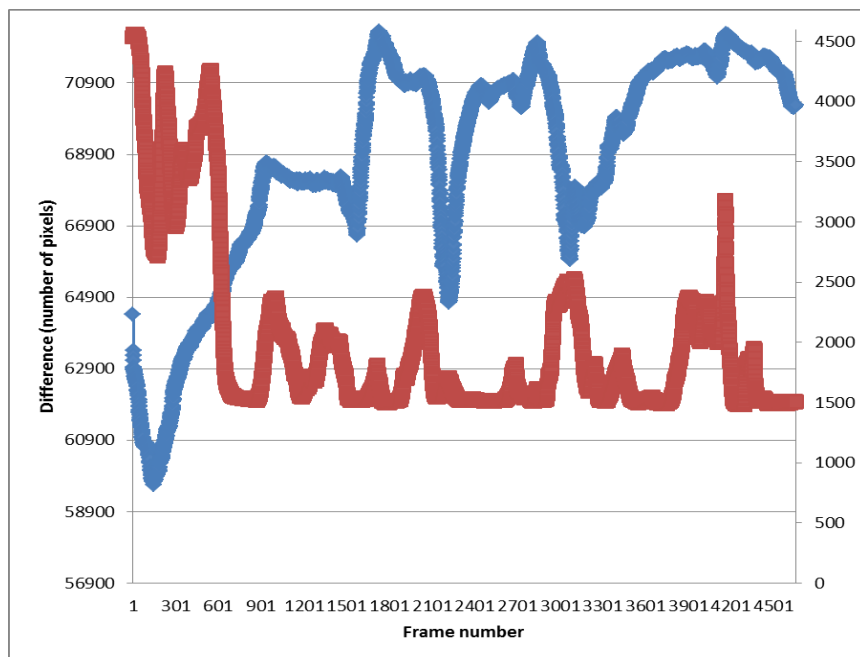


Figure 6. Difference values for GrimsonGMM using video BS2, sampling interval 1 (normal values in blue, left axis; thresholded values in red, right axis).

The reference and extracted images for LBMOG and GrimsonGMM are given in Figures 7 and 8. It can be seen that the images in GrimsonGMM are grayed (the algorithm used gray images). The movement of the rat has changed the bedding a little, as shown in the results of both algorithms. The lower left corner in GrimsonGMM has remnants of the animal left. The extracted background for LBMOG is closer to the reference background than GrimsonGMM.



Figure 7. LBMOG reference and resulting images for BS2, sampling rate 1

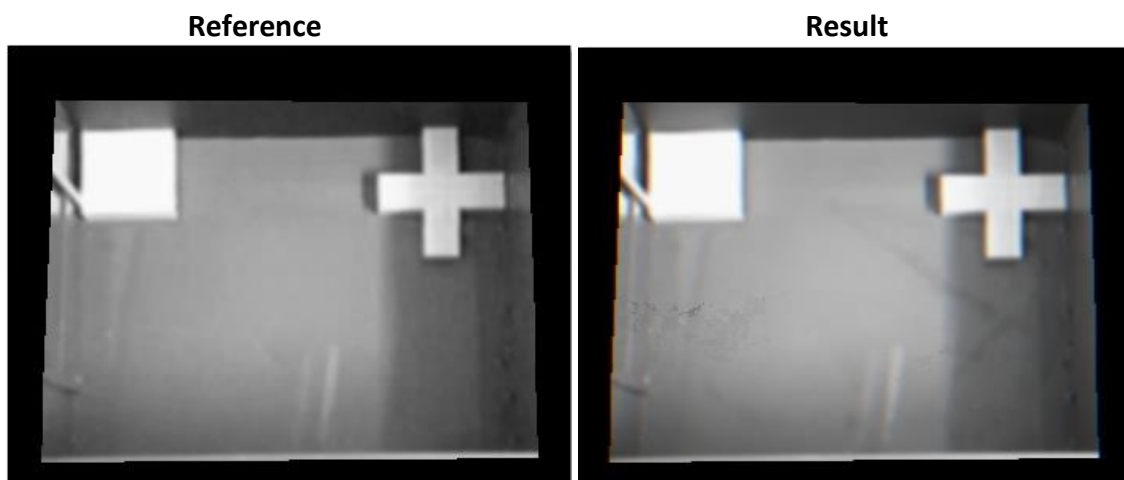


Figure 8. GrimsonGMM reference and resulting images for BS2, sampling rate 1

The examinations done above using two algorithms show that it is possible to differentiate algorithms based on Speed, Difference, and the number of occurrences of the minimum Difference values.

To summarize, the following criteria will be considered in looking for the best algorithm in the following section: Speed, minimum Difference, minimum thresholded Difference, and the frequencies of occurrence of the minimum difference values.

3. How can the evaluation be performed to get the best algorithm?

The Speed, minimum Difference, minimum “thresholded” Difference and frequencies of the minimum difference criteria selected above were incorporated in the program for the 24 different algorithms. Each algorithms was run with two videos representing different complexities and containing animals with different colors:

“BS2” video (352 x 288 resolution): contains a hooded rat in a quite complex background with hay and static objects.

Video1 video (320 x 240 resolution): contains a small active rat in an “open” field without other objects except the bedding.

For both videos, the lighting was more or less the same from beginning to end. Aside from the animal, there were no other objects added nor removed throughout the video. The images in the videos were masked so that only the arena areas were analyzed.

Both videos contained empty background at the start. The reference backgrounds were taken from this part of the video where the animal was not yet present. The algorithm runs were started at the same point in the video after the animals had already been introduced. The algorithm runs were also stopped at the same frame numbers. The durations of running each algorithm were taken as the Speed of the algorithm.

The algorithm calculated background was extracted for each frame input. The extracted and reference backgrounds were compared, and the Difference and “thresholded” Difference values were calculated. The lowest (minimum) of the difference values were taken, and the frequency of occurrence of this value was noted. The frames that correspond to the minimum values were checked to see if the extracted background did not contain traces of the animal. Using the results, it was decided whether to use the “thresholded” or “non-thresholded” Differences. The calculated values were then examined in relation to the type and modality of the algorithm, to see if there is any relation.

The last step was to perform MCE analyses using the calculated values of the different criteria. The following steps were done in the MCE analyses:

- a. Standardize the values. This was done using the following formula:

$$\text{Value}_{\text{std}} = (\text{Value}_{\text{calc}} - \text{Value}_{\text{worst}}) / (\text{Value}_{\text{best}} - \text{Value}_{\text{worst}})$$

where:

- Value_{std} : standardized value
- Value_{calc}: calculated value for the algorithm
- Value_{worst}: worst calculated value for the criteria
- Value_{best} : best calculated value for the criteria

- b. Apply the weights. The weights were determined according to the different scenarios (Speed = Quality, Speed > Quality, and Speed < Quality). For example, for the case Speed = Quality, the weight assigned to Speed was 50% and the weight assigned to Quality was 50%. The Quality weight was distributed over the minimum Difference (25%) and it’s frequency (25%).

- c. Calculate the score. The scores per criteria for each algorithm was calculated by multiplying the standardized value and the weight. Then the total score for the algorithm was computed by summing the scores of each criteria.
- d. Rank the scores. Finally, the total scores were ranked from highest (rank 1) to lowest (rank 24).

The results are discussed below for each video.

a. BS2 video

Appendix III, Figures 1 and 2 show the frames from the BS2 video with the minimum Difference and minimum “thresholded” Difference for each algorithm. It can be seen that not all algorithms show empty backgrounds. For example, Frame Difference method use the previous frame as the background frame always. For BS2, the previous frames always had animals in them. WeightedMovingMean assigns weights to 2 previous images and the current image to calculate the background, but this still resulted to backgrounds which were not empty. Few methods like FuzzyChoquetIntegral and FuzzySugenolIntegral always produced black backgrounds. From the figures, those with animals or animal traces in the background could be considered “false positives”, meaning those frames do not really give empty background.

Appendix III Figure 1 shows that none of the algorithms produced empty background using the minimum Difference measure. Appendix III Figure 2 on the other hand, shows that some algorithms produced empty backgrounds using the minimum “thresholded” Difference measure. The algorithms that produced empty backgrounds are listed in Table 3. This indicates that for these algorithms, it was necessary to remove the noise in order to get good correspondence between the extracted and reference backgrounds. The results also show that there is more than one method that can produce empty background.

Table 3. Methods which produced empty background for BS2 video using the Minimum Thresholded Difference measure.

Methods
AdaptiveBackgroundLearning
DPAdaptiveMedian
DPGrimsonGMM
DPPratiMediod
DPWrenGA
DPZivkovicAGMM
LBAadaptiveSOM
LBFuzzyGaussian
LBMixtureOfGaussians
LBSimpleGaussian

Because no empty background was found in the case with minimum Difference, this criteria was not used in further analyses. The algorithms which did not show the backgrounds (all

dark), like DPEigenbackground, FuzzyChoquetIntegral and FuzzySugenIntegral, were also not included in further analyses to increase the sensitivity of the results to the remaining algorithms.

The calculated values for Speed and minimum “thresholded” Difference were examined to see if there is any relation of the results with the type of algorithm and modality used in the algorithm. A summary of the calculated values is given in Table 4, and a graph of the results in Figure 9. High values are highlighted in the table.

Looking at the Speed results, it can be seen that except for 1 algorithm (MultiLayer_Learn), the basic types have in general faster speed (lower in number), which is followed by Statistical types, then Fuzzy types. Unimodal types are in general faster than multi-modal types.

There is no distinct relation between minimum “thresholded” Difference and algorithm type or modality. The algorithms with the lowest minimum “thresholded” Difference values are LBMixtureOfGaussians, LBFuzzyGaussians and AdaptiveBackgroundLearning, respectively. T2FGMM_UM and T2FGMM_UV have higher minimum “thresholded” Difference than the rest.

Table 4. Summary of calculated values for the different criteria for the BS2 video.

Algorithm	Type	U/M	Speed (msec)	Diff_thr (nr pixels)
AdaptiveBackgroundLearning	1	1	182,308	42
DPAdaptiveMedian	1	1	184,703	1458
DPGrimsonGMM	2	2	284,060	1477
DPMean	1	1	213,691	2364
DPPratiMediod	1	2	288,668	1482
DPWrenGA	2	1	204,972	1485
DPZivkovicAGMM	2	2	210,601	1496
FrameDifference	1	1	164,027	879
LBAadaptiveSOM	3	2	289,133	802
LBFuzzyAdaptiveSOM	3	2	309,041	1133
LBFuzzyGaussian	3	1	221,139	1
LBMixtureOfGaussians	2	2	224,250	0
LBSimpleGaussian	2	1	201,017	69
MultiLayer_Learn	2	2	544,312	81
StaticFrameDifference	1	1	161,367	2938
T2FGMM_UM	3	2	380,989	4538
T2FGMM_UV	3	2	375,522	4136
WeightedMovingMean	2	1	182,971	875

Note: Type: (1) basic, (2) statistical, (3) fuzzy
 U/M: unimodal (U) or multimodal (M)
 Diff_thr = minimum “thresholded” Difference,
 Although modality refers to Statistical and Fuzzy types, they were also considered in Basic types.

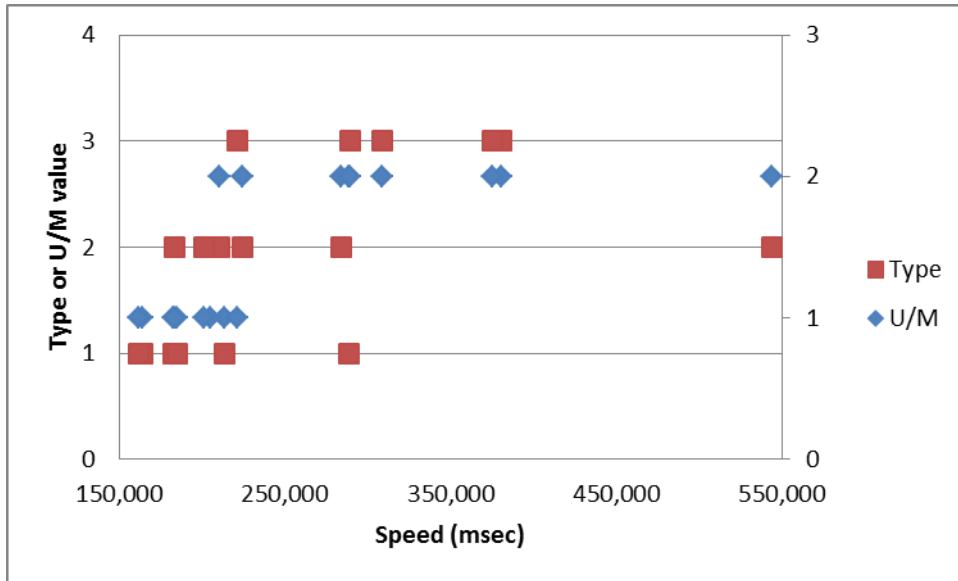


Figure 9. Relation between Speed results with Type of algorithm (left axis) and Modality (right axis) for BS2 video.

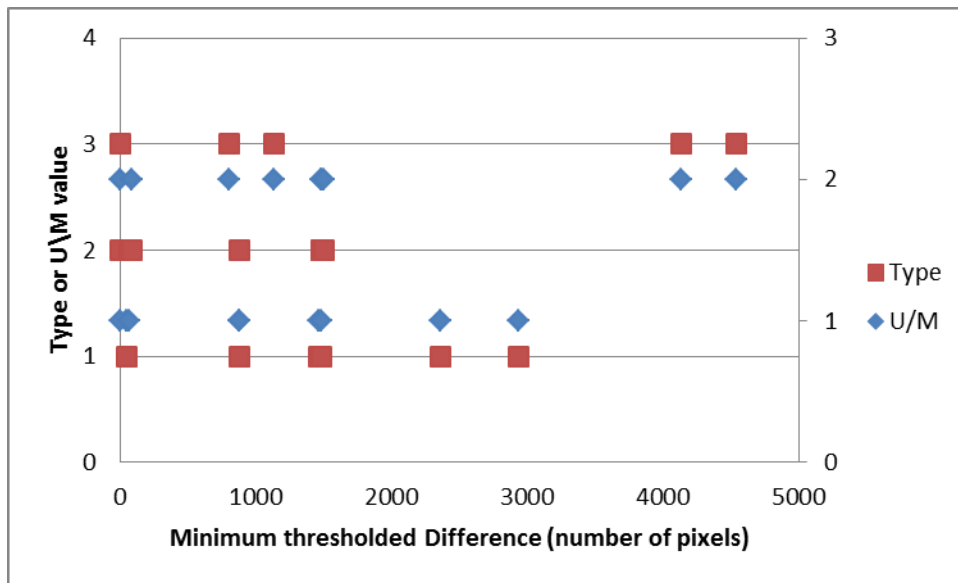


Figure 10. Relation between Quality results with Type of algorithm (left axis) and Modality (right axis) for BS2 video.

The MCE results for BS2 is given in Appendix IV. Because there were no empty backgrounds produced with minimum Difference, only the minimum “thresholded” Difference values were used. The frequency (number) of minimum “thresholded” Difference was included in the analysis. Originally, the StaticFrameDifference algorithm contained a high value for the frequency, because it used the same background (the first image in the sequence) for all the frames. The frequency for this method was set to 1. Note that Appendix IV contains 3 parts, corresponding to each scenario (or criteria as stated in the Tables).

The top scorers in the MCE analyses are as follows:

Speed = Quality and Speed > Quality:

- (1) LBMixtureOfGaussians
- (2) AdaptiveBackgroundLearning
- (3) FrameDifference

Speed < Quality:

- (1) LBMixtureOfGaussians
- (2) AdaptiveBackgroundLearning
- (3) LBSimpleGaussian

The LBMixtureOfGaussians algorithm ranked first in all the scenarios. This was influenced by the very low minimum thresholded difference value, high frequency of this value and relatively acceptable speed. Frame difference is in the top 3, for the cases when Speed was given importance, even if the background in Appendix III, Figure 2 was not empty. AdaptiveBackgroundLearning has a similar case; the extracted background still showed traces of the animal. The background were not empty because the algorithms were based on frames which always contained animals in the video. The MCE analyses could not prevent these algorithms from being chosen. Probably other factors should be included in the MCE analysis, or other preliminary steps should be considered which will exclude these algorithms in the final analysis.

Sensitivity analysis of the Frequency criteria was done to determine how the ranking of the algorithms is affected by the number of samples at a certain distance from the minimum "thresholded" Difference value. It shows how stable the rankings are over the ranges from the minimum "thresholded" Difference value. An algorithm which maintains a high rank as the range from the minimum value is increased means it's estimate of the empty background or close to it, occurs more frequently. To perform this analysis, the following frequency values were calculated: frequency of the minimum "thresholded" Difference plus 10, plus 25, plus 50 and plus 100 pixels. Table 5 shows the frequency values calculated and the Table 6 shows the top 3 (or 4) ranked algorithms using MCE analyses with Speed, minimum "thresholded" Difference, and the frequency factor. The tables show that the frequency of LBMixtureOfGaussians was the highest until minimum "thresholded" Difference plus 25 pixels. This had contributed to its first ranking in the MCE analyses. However, although the frequency of AdaptiveBackgroundLearning for the different plus factors were not high in Table 5, it still ranked high in the MCE analyses (Table 6). This could be because of good values in the other criteria (Speed and minimum "thresholded" Difference). DPZivkovicAGMM also performed well.

Table 5. Frequency values of minimum threshold difference plus different number of pixels, using BS2 video.

Algorithms	Frequency minimum thresholded difference				
	plus 0	plus 10	plus 25	plus 50	plus 100
AdaptiveBackgroundLearning	1	2	3	6	16
DPAdaptiveMedian	7	56	238	1092	2485
DPGrimsonGMM	2	50	363	1197	1918
DPMean	1	4	6	8	12
DPPratiMediod	5	45	70	255	415
DPWrenGA	1	65	203	493	1033
DPZivkovicAGMM	2	234	1213	3327	3381
FrameDifference	1	3	4	9	16
LBAadaptiveSOM	1	56	621	1899	3448
LBFuzzyAdaptiveSOM	3	119	462	863	2661
LBFuzzyGaussian	5	52	112	175	280
LBMixtureOfGaussians	768	1069	1254	1378	1530
LBSimpleGaussian	2	34	59	100	153
MultiLayer_Learn	1	5	6	8	11
StaticFrameDifference	1	1	1	1	1
T2FGMM_UM	1	17	26	29	44
T2FGMM_UV	1	3	7	19	47
WeightedMovingMean	1	2	5	15	21

Table 6. Top ranking algorithms for different scenarios and frequency values (minimum thresholded difference plus the factor), for BS2 video.

Freq	Rank	Speed = Quality	Speed > Quality	Speed < Quality
Plus 0	1	LBMixtureOfGaussians	LBMixtureOfGaussians	LBMixtureOfGaussians
	2	AdaptiveBackgroundLearning	AdaptiveBackgroundLearning	AdaptiveBackgroundLearning
	3	FrameDifference	FrameDifference	LBSimpleGaussian
Plus 10	1	LBMixtureOfGaussians	LBMixtureOfGaussians	LBMixtureOfGaussians
	2	AdaptiveBackgroundLearning	AdaptiveBackgroundLearning	AdaptiveBackgroundLearning
	3	LBSimpleGaussian	FrameDifference	LBSimpleGaussian
Plus 25	1	LBMixtureOfGaussians	LBMixtureOfGaussians	LBMixtureOfGaussians
	2	DPZivkovicAGMM	DPZivkovicAGMM	DPZivkovicAGMM
	3	AdaptiveBackgroundLearning	AdaptiveBackgroundLearning	AdaptiveBackgroundLearning
Plus 50	1	DPZivkovicAGMM	DPZivkovicAGMM	DPZivkovicAGMM
	2	LBMixtureOfGaussians	LBMixtureOfGaussians	LBMixtureOfGaussians
	3	DPAdaptiveMedian	AdaptiveBackgroundLearning	LBAadaptiveSOM
Plus 100	1	DPZivkovicAGMM	DPZivkovicAGMM	DPZivkovicAGMM
	2	DPAdaptiveMedian	DPAdaptiveMedian	LBAadaptiveSOM
	3	LBAadaptiveSOM	LBMixtureOfGaussians	DPAdaptiveMedian
	4	LBMixtureOfGaussians	AdaptiveBackgroundLearning	LBMixtureOfGaussians

b. Video1 video

The same analysis was done with the Video1 video. For the runs, some changes were made in some algorithms because of the number of frames that were accessible from the video. For example, for FuzzySugenIntegral and FuzzyChoqueIntegral the initial number of frames used for learning were initially 200. They were both limited to 10 because of the limited number of frames in Video1. For LBMixtureOfGaussians, the learning rate was change from 59 to 80 to account for the fast mouse. Appendix V shows frames from the video that corresponds with the minimum Difference (Figure 1) and minimum “thresholded” Difference (Figure 2) for each algorithm. Looking at the frames, both DPPratiMediod and LBMixtureOfGaussians showed empty backgrounds in both Figures. MultiLayer also looked empty in Figure 2 (minimum “thresholded” Difference).

To be comparable with the results for BS2, the results for minimum Difference were not included in further analyses. The algorithms which did not show the backgrounds well (black), like DPEigenbackground, FuzzyChoquetIntegral and FuzzySugenIntegral, were also not included in further analyses to increase the sensitivity of the results to the remaining algorithms.

The calculated Speed and Quality values were examined in relation to the type of algorithm and the modality used in the algorithm. The results are given in Table 7 and Figures 11 and 12. High values are highlighted in the Table.

Looking at the Speed results, it can be seen that except for 2 algorithms (DPPratiMideod and MultiLayer_Learn), the basic types have in general faster (lower duration), which is followed by Statistical types, then the Fuzzy types. Unimodal types are in general faster than multi-modal types.

There is no distinct relation between minimum “thresholded” Difference and algorithm type nor modality. Low values of minimum “thresholded” Difference can be found in LBMixtureOfGaussians, LBAadaptiveSOM and LBFuzzyAdaptiveSOM. High values can be found in DPGrimsonGMM, DPZivkovicAGMM, and T2FGMM_UV.

The MCE analyses applied to Video1 is given in Appendix VI. The frequency for StaticFrameDifference which uses the first frame as background for succeeding frames, was set to 1. To make the analysis similar to BS2, Appendix VI contains minimum thresholded Difference values.

Table 7. Summary of calculated values for the different criteria for the Video1 video.

Algorithm	Type	U/M	Speed (msec)	Diff_thr (nr pixels)
AdaptiveBackgroundLearning	1	1	6,750	158
DPAadaptiveMedian	1	1	7,060	872
DPGrimsonGMM	2	2	7,553	901
DPMean	1	1	7,327	826
DPPratiMediod	1	2	9,960	473
DPWrenGA	2	1	7,265	851
DPZivkovicAGMM	2	2	7,199	900
FrameDifference	1	1	7,255	353
LBAadaptiveSOM	3	2	7,852	38
LBFuzzyAdaptiveSOM	3	2	8,026	90
LBFuzzyGaussian	3	1	7,473	423
LBMixtureOfGaussians	2	2	7,286	0
LBSimpleGaussian	2	1	7,188	425
MultiLayer_Learn	2	2	11,660	11
StaticFrameDifference	1	1	6,686	429
T2FGMM_UM	3	2	7,805	887
T2FGMM_UV	3	2	7,544	906
WeightedMovingMean	2	1	7,373	377

Note: Type: (1) basic, (2) statistical, (3) fuzzy

U/M: unimodal (U) or multimodal (M)

Diff_thr = minimum "thresholded" Difference,

Although modality refers to Statistical and Fuzzy types, they were also considered in Basic types.

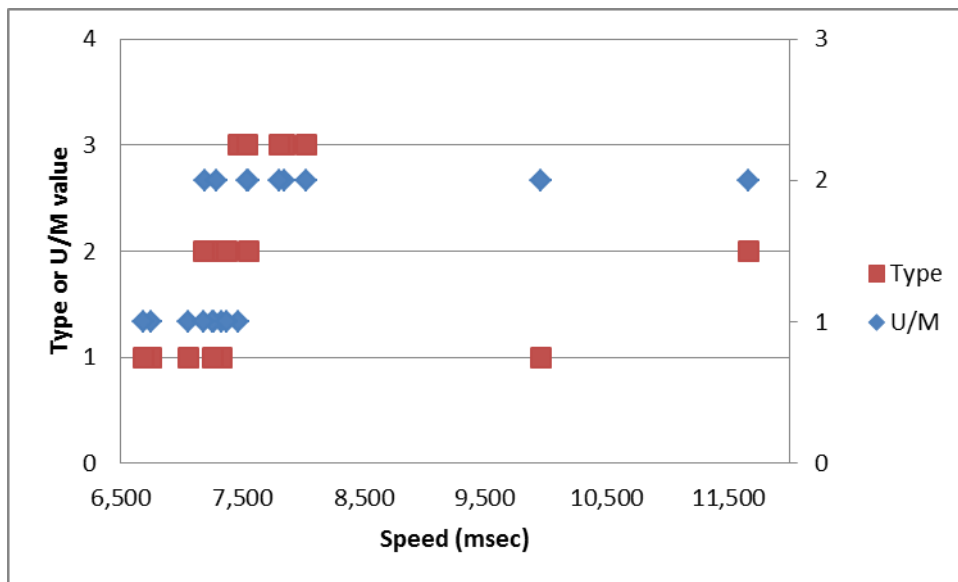


Figure 11. Relation between Speed results with Type of algorithm (left axis) and Modality (right axis) for Video1 video.

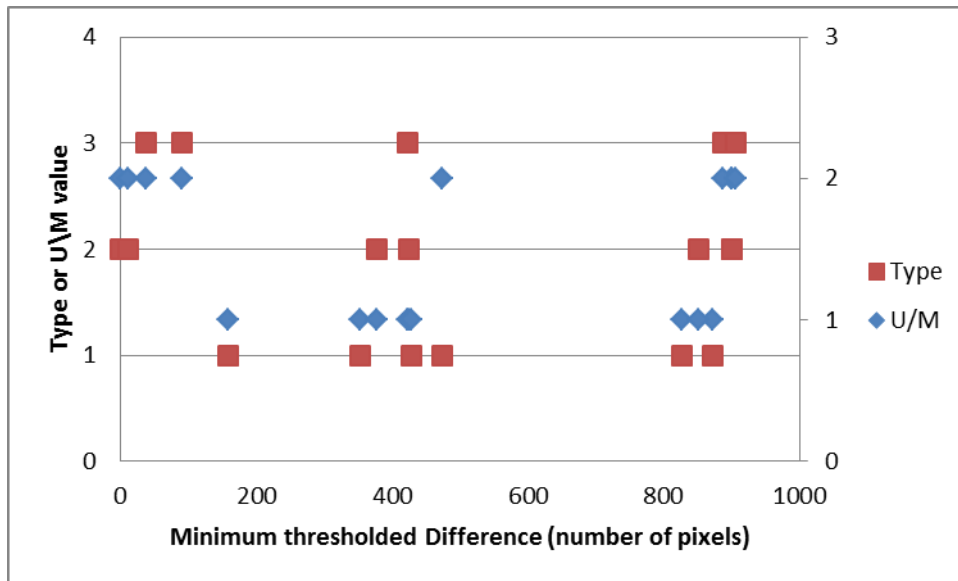


Figure 12. Relation between Quality results with Type of algorithm (left axis) and Modality (right axis) for Video1 video.

The top scorers in Appendix VI are:

Speed = Quality:

- (1) LBMixtureOfGaussians
- (2) AdaptiveBackgroundLearning
- (3) DPAdaptiveMedian

Speed > Quality:

- (1) LBMixtureOfGaussians
- (2) AdaptiveBackgroundLearning
- (3) StaticFrameDifference

Speed < Quality:

- (1) LBMixtureOfGaussians
- (2) AdaptiveBackgroundLearning
- (3) LBAadaptiveSOM

From the results, the LBMixtureOfGaussians scored the best in all scenarios. Just like with BS2, there were two algorithms, the AdaptiveBackgroundLearning and StaticFrameDifference, which came to the top 3 but which showed the animal or traces of it in the extracted background (Appendix VI, Figure 2). This was due to the effect of Speed in the MCE result.

Just like with BS2, sensitivity analysis of the Frequency criteria was performed to determine how the ranking of the algorithms is affected by the number of samples at a certain distance from the minimum “thresholded” Difference value.. Table 8 shows the frequency values calculated and the Table 9 shows the top 3 ranked algorithms using MCE analyses with Speed, minimum “thresholded” Difference, and the frequency factor. The tables show that the frequency of

LBMixtureOfGaussians was the highest only in Plus 0 (before minimum “thresholded” Difference value + 10 pixels). Other algorithms like DPGrimsonGMM, DPZivkovicAGMM, LBFuzzyGaussian and LBSimpleGaussian had all their samples included in the range minimum “thresholded” Difference plus 10 pixels. This is the reason why, in the MCE analyses which also considered Speed and minimum “thresholded” Difference, LBSimpleGaussian and LBFuzzyGaussian also ranked well (Table 9). Still, the results show that LBMixtureOfGaussians performed in the different ranges well.

Table 8. Frequency values of minimum threshold difference plus different number of pixels, using Video1 video.

Algorithms	Frequency minimum thresholded difference				
	plus 0	plus 10	plus 25	plus 50	plus 100
AdaptiveBackgroundLearning	1	1	1	3	4
DPAdaptiveMedian	7	28	61	62	62
DPGrimsonGMM	2	62	62	62	62
DPMean	1	1	2	4	23
DPPratiMediod	5	31	36	41	41
DPWrenGA	2	13	42	57	62
DPZivkovicAGMM	1	62	62	62	62
FrameDifference	1	3	3	4	28
LBAadaptiveSOM	1	2	3	3	3
LBFuzzyAdaptiveSOM	1	2	2	8	14
LBFuzzyGaussian	1	62	62	62	62
LBMixtureOfGaussians	10	27	31	32	33
LBSimpleGaussian	1	62	62	62	62
MultiLayer_Learn	1	28	35	38	43
StaticFrameDifference	1	1	1	1	1
T2FGMM_UM	1	15	46	60	62
T2FGMM_UV	2	24	28	62	62
WeightedMovingMean	1	1	5	12	34

Table 9. Top ranking algorithms for different scenarios and frequency values (minimum thresholded difference plus the factor), for Video1 video.

Freq	Rank	Speed = Quality	Speed > Quality	Speed < Quality
Plus 0	1	LBMixtureOfGaussians	LBMixtureOfGaussians	LBMixtureOfGaussians
	2	AdaptiveBackgroundLearning	AdaptiveBackgroundLearning	AdaptiveBackgroundLearning
	3	DPAdaptiveMedian	StaticFrameDifference	LBAadaptiveSOM
Plus 10	1	LBSimpleGaussian	LBSimpleGaussian	LBSimpleGaussian
	2	LBFuzzyGaussian	LBMixtureOfGaussians	LBFuzzyGaussian
	3	LBMixtureOfGaussians	LBFuzzyGaussian	LBMixtureOfGaussians
Plus 25	1	LBSimpleGaussian	LBSimpleGaussian	LBSimpleGaussian
	2	LBMixtureOfGaussians	LBMixtureOfGaussians	LBMixtureOfGaussians
	3	LBFuzzyGaussian	LBFuzzyGaussian	LBFuzzyGaussian
Plus 50	1	LBSimpleGaussian	LBSimpleGaussian	LBSimpleGaussian
	2	LBMixtureOfGaussians	LBMixtureOfGaussians	LBMixtureOfGaussians
	3	LBFuzzyGaussian	LBFuzzyGaussian	LBFuzzyGaussian
Plus 100	1	LBSimpleGaussian	LBSimpleGaussian	LBSimpleGaussian
	2	LBMixtureOfGaussians	LBMixtureOfGaussians	LBMixtureOfGaussians
	3	LBFuzzyGaussian	LBFuzzyGaussian	LBFuzzyGaussian

The overall result of the MCE is reasonable in the sense that the best scorer, the LBMixtureOfGaussians consistently showed empty background in the frames corresponding to the minimum “thresholded” Difference. The LBMixtureOfGaussians algorithm also consistently scored high when different frequency ranges from the minimum “thresholded” Difference were considered. But what is quite strange in the results is that sometimes an algorithm which shows the animal in the extracted background, like FrameDifference, can get a high score. This could mean that the criteria put in the MCE analyses were not good. Another implication is that the Speed and Quality measures may not have been well defined. Speed could, for example, have been defined by taking the time until the first empty image was extracted. Also, a minimum Difference can occur when the animal is standing (less pixel area used). Moreover, the difference value could have been caused by other factors, like noise, and not only by the presence of the animal in the background. Another way of dealing with this problem is probably to give Speed, which is the big reason why Frame Difference ranked high, less weight. In any case, taking all results together, LBMixtureOfGaussians can reasonably be considered as the best algorithm.

The LBMixtureOfGaussians method will be used for the rest of the study. Its source code can be downloaded from Sobral (2013a).

4. How does the algorithm perform in different laboratory situations

The best algorithm was used with different videos to check if it can extract empty backgrounds. Of the default settings in the program, only the Learning rate setting was changed from 59 to 80 to accommodate to fast moving animals.

The program was run for 37 different videos, representing different animals, animal size and color, different background (constant and non-constant, water, beddings, with reflections, poor lighting, objects in arena, etc). There were experiments with single and multiple arenas, and single and multiple animal in one case. The set also includes videos for which the algorithm is known to fail (animal hardly move).

The results for the different videos are shown in Appendix VII, Figure 1. The last result for 96 arenas is enlarged in Appendix VII, Figure 2, to show the arena contents better. The first column in the Appendix VII Figure 1 shows the name and resolution of the videos. The second column shows the arenas with animal. The third column shows the extracted empty background. The last column contains some comments on the results.

As can be seen in the images, an empty background is extracted in most situations. Exceptions to this are in BD2, BD6, BM4, BM5, BM6 and BS11. In the first five of these videos (e.g. except for BS11), the animals hardly moved so that a part (or whole) of the animal was left in the extracted backgrounds. For BS11, the run was stopped when a considerable number of arenas were emptied. This is because for multiple arenas, especially when the number of arenas is more than 4, it was observed that the animals moved in different ways and speeds. It was sometimes difficult to wait for the time when all arenas did not contain traces of the animal anymore.

From the above results, it can be said that the algorithm fulfills the other criteria for a good method, as stated in Section IV A. It can successfully handle most of the situations in the laboratory:

1. It works for different type of animals: fast or slow moving, small or large, one or more animals, one or more-colored animal. However, the algorithm cannot estimate the background when the animal hardly moves. But this is acceptable since in the assumptions in Section IV A, it was said that: a foreground object can remain stationary for a short interval, but no longer than the interval where the background is revealed. So for animals that hardly move, the background should be taken before putting the animal in the cage.
2. It works for different types of setups: one arena or multiple arena setup, open field, Y maze, Plus maze, etc. However, for some multiple arena cases, it was difficult to get empty background for all arenas, especially when the animals moved differently. It is probably good to run the algorithm a second time for only the arena's that are left with part of the animal present.
3. It works when there is no animal at the start of the run.
4. It deals or tolerate the changes affecting the background which are not caused by the foreground object like change in lighting, shadows cast, and moving background (changes in the bedding or ripples in the water caused by movement of the animal).

It is possible to influence the results by changing the values of the parameters used in the model.

V. Summary and Conclusions

The study identified characteristics of a good algorithm for background extraction to be fast (speed), of good quality (produce empty background) and applicability (can be used for different situations in the laboratory). The internet was sought for available algorithms, and background subtraction algorithms which contain background modeling components were used to extract the background. The algorithms came with a range of characteristics, like types, modality, data abstraction level, feature size, etc. Studying each algorithm suggested that there are a number of them which can produce empty backgrounds and fulfil the desired characteristics of a good algorithm.

Different quantifiable criteria were selected to use for selecting the best algorithm. To make good comparisons among different algorithms, these criteria were supposed to be independent of the characteristics of the algorithms used. The criteria identified were Speed, minimum Difference, and the frequency of occurrence of the minimum Difference value. The last two (except for Speed) were measurements of quality. They were calculated by comparing extracted backgrounds from the algorithms and reference background. The calculations were done on normal and thresholded images. Test of the the criteria on two algorithms showed that they could differentiate the performances of the algorithms.

When the criteria was applied to all the algorithms, it showed that for some algorithms, the frames corresponding to minimum Difference or minimum “thresholded” Difference did not always show empty backgrounds. When the criteria were examined against the type of method and modality of the method, it showed that Speed is in general related to type and modality of the method, but the other criteria not. This means that Speed as criteria was not really independent of the algorithms. MCE analyses were performed for the scenarios that considered different preferences of users, like Speed = Quality, Speed > Quality and Speed < Quality. The results showed that LBMixtureOfGaussians (LBMOG) was the best algorithm for the videos used. LBMOG is based on the Gaussian distribution. It is multi-modal, parametric, recursive and adapts to changes in the background.

The LBMOG algorithm was used to extract the background of videos representing different situations. Empty background was obtained in many cases. Exceptions were in videos where the animal hardly moved. It also did not perform well in multiple arenas, because it was difficult to find the same moment of time when the arenas were all empty.

LBMOG answers the requirements on the characteristics of a good algorithm for laboratory use, as was initially defined in the study. It is applicable in many situations, has low computational and memory requirement, and is relatively fast. It contains settings, whose values can be changed to improve the performance of the algorithm. It is recommended for use in the laboratory situation.

However, there were some results which were not as expected. For example, FrameDifference, StaticFrameDifference, and AdaptiveBackgroundLearning, which always had backgrounds with animal or traces of animal, got relatively high scores in the MCE analysis. This could mean that the Speed criteria which was related to the type of algorithm, should probably not have been added as criteria in the MCE analyses, or should have gotten less weight, or should have been defined in another way. Another implication is that Quality measures could have been not very well defined. The minimum Difference, for example, compare the intensities of each pixel and

gets the total number of pixels with differences. However, the differences could have been caused by various factors, and not only by the presence of the animal in the extracted background. The cut off value for thresholding should probably have been higher.

For future study, it is suggested to:

1. Examine the MCE analysis method and devise a way to prevent algorithms that cannot produce empty backgrounds from ending up with high scores. The frequencies of the values with minimum difference was included to do this task, but still some of these algorithms ended up with high scores.
2. Optimize the extraction of the final background. The algorithms considered here run from start to end. User intervention is needed to stop the algorithm when an empty background is obtained. It would be good if the algorithm themselves are able to stop once an empty background is obtained.

Lastly, from literature review, it was realized that many studies have already been made, and algorithms created to extract empty backgrounds. Many of the reported studies are without source codes, but there are authors who are willing to share their programs. There are good algorithms which work well for many situations, but not for all situations. For this case, it is handy to be able to change parameters in the algorithm to tune it better to a particular application.

References

- Bender, L. (2013). "Background Subtraction Models." Retrieved 16 June, 2013, from <http://scene.sourceforge.net/models.html>.
- Benezeth, Y., Jodoin, P.M., Emile, B., Laurent, H., and Rosenberger, C. (2008). "Review and Evaluation of Commonly-Implemented Background Subtraction Algorithms." IEEE International Conference on Pattern Recognition: 1-4.
- Bouwmans, T., El Baf, F., and Vachon, B. (2008). "Background Modeling using Mixture of Gaussians for Foreground Detection – A Survey." Recent Patents on Computer Science **1**(3): 219-237.
- Chen, T. P., Haussecker, H., Bovyrin, A., Belenov, R., Rodyushkin, K. Kuranov, A., and Eruhimov, V. (2005). "Computer Vision Workload Analysis: Case Study of Video Surveillance Systems." Inter Technology Journal **9**(2).
- Cheung, S.-C., and Kamath, C. (2004). "Robust Techniques for Background Subtraction in Urban Traffic Video." SPIE Electronic Imaging - Video Communications and Image Processing: 881–892.
- Cohen, I and Medioni, G. (1999). "Detecting and tracking moving objects for video surveillance." IEEE Proc. Computer Vision and Pattern Recognition, Jun 23-25, 1999.
- Criminisi, A., Perez, P., and Toyama, K. (2004). "Region Filling and Object Removal by Exemplar-Based Image Inpainting." IEEE Transactions on Image Processing **13**(9).
- Cristani, M., Bicego, M., and Murino, V. (2003). "Multi-level background initialization using Hidden Markov Models." First ACM SIGMM International Workshop on Video Surveillance.
- Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. (2003). "Detecting moving objects, ghosts, and shadows in video streams." IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(10): 1337-1342.
- Eghbali, H. J. (1979). "K-s test for Detecting Changes from Landsat imagery data." IEEE Transactions on Systems, Man and Cybernetics **9**(1): 17–23.
- El Baf, F., Bouwmans, T., and Vachon, B. (2008a). "Fuzzy Integral for Moving Object Detection." FUZZ-IEEE 2008, Hong Kong.
- El Baf, F., Bouwmans, T., and Vachon, B. (2008b). "Type-2 Fuzzy Mixture of Gaussians Model: Application to Background Modeling." ISVC 2008, Las Vegas, USA.
- Elgammal, A. (2011). "Figure-Ground Segmentation – Pixel-Based." Visual Analysis of Humans, T.B. Moeslund et al. (Eds), Springer-Verlag London Limited 2011.
- Elhabian, S. Y., El-Sayed, K.M., and Ahmed, S.H. (2008). "Moving Object Detection in Spatial Domain using Background Removal Techniques - State-of-Art." Recent Patents on Computer Science **1**: 32–54.
- Freedman, N., and Russell, S (1997). "Image segmentation in video sequences: A probabilistic

approach." Conference on Uncertainty in Artificial Intelligence.

Gutchess, D., Trajkovic, M., Cohen-Solal, E., Lyons, D., and Jain, A.K. (2001). "A Background Model Initialization Algorithm for Video Surveillance." Eight International Conference on Computer Vision **1**: 733-740.

Heywood, I., Cornelius, S., and Carver, S. (2006). An Introduction to Geographical Information Systems. England, Pearson Education Limited.

Jain, R. C., and Nagel, H.H. (1979). "On the analysis of accumulative difference pictures from image sequences of real world scenes." IEEE Transactions on Pattern Analysis and Machine Intelligence **1**(2): 206–214.

KaewTraKulPong, P., and Bowden, R. (2001). "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection." Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01, September 2001.

Madalena, L., and Petrosino, A. (2008). "A Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications." IEEE Transactions on Image Processing **17**(7): 1168-1177.

Madalena, L., and Petrosino, A. (2009). "Multivalued Background/Foreground Separation for Moving Object Detection." WILF 2009, LNAI 5571, DiGesù, V., Pal, S.K., and Petrosino, A. (Eds) 263-270.

Madalena, L., and Petrosino, A. (2010). "A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection." Neural Computing and Applications **19**: 179-186.

McFarlane, N. J. B., and Schofield, C.P. (1995). "Segmentation and tracking of piglets in images." Machine Vision and Applications **8**: 187-193.

Noldus, L. P., Spink, A.J., and Tegelenbosch, R.A. (2001). "EthoVision: a versatile video tracking system for automation of behavioral experiments." Behavior Research Methods, Instruments & Computers **33**(3): 398–414.

Oliver, N. M., Rosario, B., and Pentland, A.P. (2000). "A Bayesian Computer Vision System for Modeling Human Interactions." IEEE Transactions On Pattern Analysis and Machine Intelligence **22**(8): 831-843.

OpenCV (2013). Retrieved 19 November, 2012, from <http://opencv.org/>.

Piccardi, M. (2004). "Background Subtraction Techniques: a Review." IEEE International Conference on Systems, Man and Cybernetics: 3099–3104.

Radke, R. J., Andra, S., Al-Kofahi, O., and Roysam, B. (2005). "Image Change Detection Algorithms: A Systematic Survey." IEEE Transactions on Image Processing **14**: 294–307.

Reddy, V., Sanderson, C., Lovell, B., and Bigdeli, A. (2009). "An efficient background estimation algorithm for embedded smart cameras." Distributed Smart Cameras, ICDCS 2009, Third ACM/IEEE International Conference.

Reddy, V., Sanderson, C., and Lovell, B. (2011). "A low-complexity algorithm for static background estimation from cluttered image sequences in surveillance contexts." EURASIP Journal on Image and

Video Processing, Volume 2011, 13 pages.

Sigari, M. H., Mozayani, N., and Pourreza, H. R. (2008). "Fuzzy running average and fuzzy background subtraction: concepts and application." IJCSNS International Journal of Computer Science and Network Security **8**(2): 138-143.

Sobral, A. (2012). "BGSLibrary." Retrieved 19 November, 2012, from <http://code.google.com/p/bgslibrary/>.

Sobral, A. (2013). "BGSLibrary: An OpenCV C++ Background Subtraction Library." IX Workshop de Visão Computacional. Rio de Janeiro, Brazil, June 2013.

Sobral, A. (2013a). "BGSLibrary: A Background Subtraction Library. Downloads". Retrieved 19 November, 2012, from <http://code.google.com/p/bgslibrary/downloads/list>.

Stauffer, C., and Grimson, W.E.L. (1999). "Adaptive background mixture models for real-time tracking." IEEE Conference on Computer Vision and Pattern Recognition, Volume 2.

Varadarajan, S., Karan, L., and Florencio, D. (2009). "Background Recovery from Video Sequences using Motion Parameters." IEEE International Conference on Acoustics, Speech and Signal Processing : 989-992.

Wang, H., and Suther, D. (2005). "Background initialization with a new robust statistical approach." IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance : 153-159.

Wren, C. R., Azerbayejani, A., Darrell, T., and Pentland, A.P. (1997). "Pfinder: Real-Time Tracking of the Human Body." IEEE Transactions on Pattern Analysis and Machine Intelligence **19**(7): 780-785.

Yao, J., and Odobez, J.M. (2007). Multi-Layer Background Subtraction Based on Color and Texture. IEEE Computer Vision and Pattern Recognition Conference. Minneapolis, MN.

Zhang, H., and Xu, D. (2006). "Fusing color and texture features for background model." Fuzzy Systems and Knowledge Discovery 2006, Lecture Notes in Computer Science **4223** : 887-893.

Zivkovic, Z., and van der Heijden, F. (2006). "Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction." Pattern Recognition Letters **27**(7): 773-780.

Appendix I. Algorithms used to extract the background

Method	Type	U/MP/N	Feature		R/N	Description of algorithm or paper which was the basis of the algorithm	
			Type	Size			
AdaptiveBackgroundLearning	Basic	-	-	intensity	frame	R	adapts background model using temporal blending or exponential forgetting, where the current background is made equal to $img_background_t = \alpha * img_input + (1-\alpha)*img_background_{t-1}$; where alpha controls the speed of forgetting the information (Elgammal, 2011)
DPAdaptiveMedian	Basic	-	-	blob edges	pixel	N	employs image differencing with respect to a median background and a Laplacian operator; uses a reference background image which is calculated using the running median of the image sequence; uses a mask to differentiate stationary objects from the background; the algorithm performed good in distinguishing the animals from the background (McFarlane and Schofield, 1995)
DPEigenbackground	Statistical	M	N	color, texture, or others	pixel, blobs	N	detects and tracks moving objects and outputs a feature vector describing motion and heading of the moving object and spatial relationship to other moving objects; the feature vectors are the inputs to a stochastic state-based behavior model and the behaviors are classified; uses an eigenspace model that is formed by computing the mean background image and its covariance matrix from N samples, the covariance matrix is diagonalized through an eigenvalue decomposition; only M largest eigenvalues are kept in the principal component analysis (PCA) for the eigenbackgrounds; the background is modeled by projecting each input image onto the space expanded by the eigenbackground images and their means (Oliver et al., 2000)
DPGrimsonGMM	Statistical	M	P	color	pixel	R	models values of each pixel as a mixture of Gaussians; pixels that correspond to background distributions are considered background; adapts robustly to lighting changes, repetitive motions in the background, cluttered regions slow moving objects, and introduction/removal of objects from the background; applied in different environments and animals (Stauffer and Grimson, 1999)
DPMean	Basic	-	-	color	pixel	N	for each pixel and channel in the image, the background is updated by taking the half of the mean; the mean is calculated by $mean = \alpha * previous\ mean + (1-\alpha) * img_input$, where alpha controls the speed of forgetting the information; a pixel is considered to be from the background if the squared distance between it and the background model is less than the threshold
DPPratiMediod	Basic	-	-	color, motion	pixel	N	uses an object-level classification of moving objects into moving visual objects (MVOs), ghosts and shadows; uses motion and shadow information to exclude MVOs and shadows, while retaining ghosts in the background; differentiates shadows by analyzing chromaticity in the HSV color space; background update is done by taking the median value between values in previous frames and the current background (Cucchiara, et al., 2003)

Following abbreviations were used: U/M = unimodal or multi-modal; P/N =- parametric or non-parametric; R/N = recursive or non-recursive

Method	Type	U/MP/N	Feature			R/N	Description of algorithm or paper which was the basis of the algorithm
			Type	Size			
DPWrenGA	Statistical	U	P	color	pixel, blobs	R	originally used for tracking people and interpreting their behavior (Pfinder software), but the algorithm can be applied to track vehicles and animals; adopts a Maximum A Posteriori Probability (MAP) approach in detection and tracking using 2D models; incorporates a priori knowledge on the foreground in bootstrapping and recovering from errors; feature vectors at each pixel are clustered to form blobs, which are coherent, connected regions; models background as a texture surface with each point in the surface associated with mean color value and color distribution modeled with Gaussian distribution (Wren, et al., 1997).
DPZivkovicAGMM	Statistical	M	P	color	pixel	R	similar to MixtureOfGaussianV2, but used Donovan Parks (see Sobral, 2012) own C++ implementation
FrameDifference	Basic	-	-	gray or color	frame	N	uses previous frame as background and calculates the absolute difference between the background and the current image as the foreground
FuzzyChoquetIntegral	Fuzzy	M	N	color, edge, texture, stereo	pixel	R	fuses color and texture features using Choquet integral as aggregating operator; transforms RGB color space to YCrCb color space and uses local binary pattern operator for the texture model; uses similarity measures for color and texture, which are then aggregated by the Choquet integral; uses a background update rule that quickly adapts a pixel classified as background and slowly adapts a pixel that is classified as foreground; robust to changes happening in the background (El Baf, et. al., 2008a)
FuzzySugenolIntegral	Fuzzy	M	N	color, texture	pixel	R	fuses color and texture features using fuzzy integral; transforms RGB color space to Ohta color space and uses local binary pattern operator for the texture model; uses pixel motion character to decide if the pixel has to be updated to the background or not; robust to changes in the background (Zhang and Xu, 2006)
LBAadaptiveSOM	Fuzzy	M	N	neural map	pixel	R	uses competitive neural network similar to Kohonen Self-Organizing Map (SOM) to adaptively model the background; a neuronal map of 3x3 vectors is defined for each pixel; incoming source pixels are mapped to the closest weight vectors according to Euclidean distance metric, and the weights are updated; the set up weights act as background model (Bender, 2013)
LBFuzzyAdaptiveSOM	Fuzzy	M	N	neural map	pixel	R	modified version of LBAadaptiveSOM; uses fuzzy rule for neural network background model update; more robust to illumination changes compared to LBAadaptiveSOM (Bender, 2013, and Madalena and Petrosino, 2008)
LBFuzzyGaussian	Fuzzy	U	P	color	pixel	R	modified version of LBSimpleGaussian using fuzzy classification rule and fuzzy online cumulative averages; provides better segmentation of stationary foreground objects than the simple Gaussian model (Bender, 2013, and Sigari, et al., 2008)

Following abbreviations were used: U/M = unimodal or multi-modal; P/N =- parameteric or non-parametric; R/N = recursive or non-recursive





Method	Type	U/MP/N	Feature			R/N	Description of algorithm or paper which was the basis of the algorithm
			Type	Size			
LBMixtureOfGaussians	Statistical	M	P	color	pixel	R	similar to LBSimpleGaussian except that this uses classic multivariate Gaussian mixture model representing each pixel by a mixture of 4 Gaussians (Bender, 2013, and Bouwmans, et al. 2008)
LBSimpleGaussian	Statistical	U	P	color	pixel	R	uses single Gaussian probability density function based on recent pixel values; updates mean and covariance matrix using online cumulative average; calculates Mahalanobis distance between source and background pixels and compares this to a threshold (Bender, 2013, Benezeth, et al., 2008)
MixtureOfGaussianV1	Statistical	M	P	color	pixel	R	uses OpenCV's cv::BackgroundSubtractorMOG class based on KaewTraKulPong and Borden (2001); improved the update mechanism in the algorithm of Stauffer and Grimson (1999) leading to faster and more accurate learning phase and improved shadow detection.
MixtureOfGaussianV2	Statistical	M	P	color	pixel	R	uses OpenCV's cv::BackgroundSubtractorMOG2 class based on Zivkovic and van der Heijden (2006); improved the algorithm of Stauffer and Grimson (1999) by automatically updating the parameters of the model and selecting the needed number of components per pixel to fully adapt to the observed scene.
MultiLayer_Learn	Statistical	M	N	color, texture	pixel	R	uses photometric invariant color measurements in RGB color space and texture features represented by local binary pattern; using simple layer based strategy, moving background pixels are modelled using quasi-periodic flickering; robust to changes in the background (Yao and Odobez, 2007)
StaticFrameDifference	Basic	-	-	gray or color	frame	N	similar to FrameDifference only it uses first frame as background.
T2FGMM_UM	Fuzzy	M	P	color	pixel	R	this is an improvement to the mixture of gaussians model to handle dynamic changes in the background and false initialization; uses T2 membership functions to represent uncertainty in the mean vector of the multivariate Gaussian model (El Baf, et al., 2008b)
T2FGMM_UV	Fuzzy	M	P	color	pixel	R	similar to T2FGMM_UM, only the T2 membership functions represent uncertainty in the variance vector of the multivariate Gaussian model; T2GMM_UM is a better estimator than T2GMM_UV (El Baf, et al., 2008b)
WeightedMovingMean	Basic	-	-	intensity	pixel	N	uses weighted average to model the background; the current image gets 50% weight, the previous image 30% and the pre-previous image 20%; if weights are not used, then uses simple average of the three images
WeightedMovingVariance	Basic	-	-	intensity	pixel	N	similar to WeightedMovingMean, but also calculated weighted variance using the same proportions as the mean; uses the square root of the total variance to estimate the foreground.

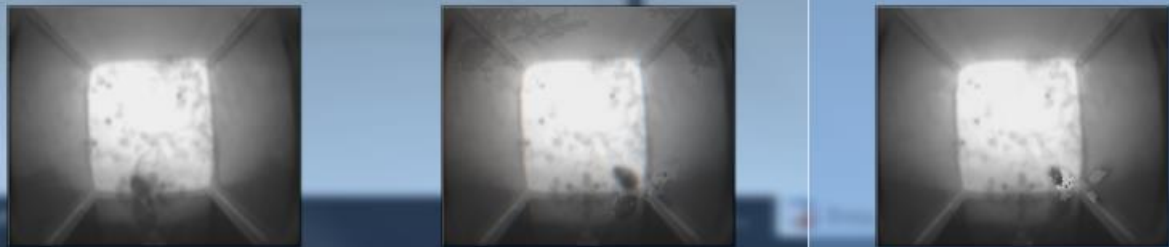

Following abbreviations were used: U/M = unimodal or multi-modal; P/N = parameteric or non-parameteric; R/N = recursive or non-recursive

Appendix II. Comparison of GrimsonGMM and LBMOG using different videos.

Note: The SInt = sample interval (frames), SR = video sample rate; the status of change is indicated as Dissappear (D) or Appear (A).

No	Video	SInt	Start image	GrimsonGMM	LBMOG	D/A
1	Green	1				D
2	Green	1				A
3	BM1	1				A
4	BS8	1				D

5	BS8	1		A
6	BS9	1		D
7	BS9	1		A
8	BD15	1		D

9	BS2	1		A
10	BD4	1		D
11	BD4	10		D
12	BD4	SR		D

Appendix III. Frames corresponding to quality measures for different background extraction methods using BS2 video.

Figure 1. Minimum Difference Frames for BS2 video.

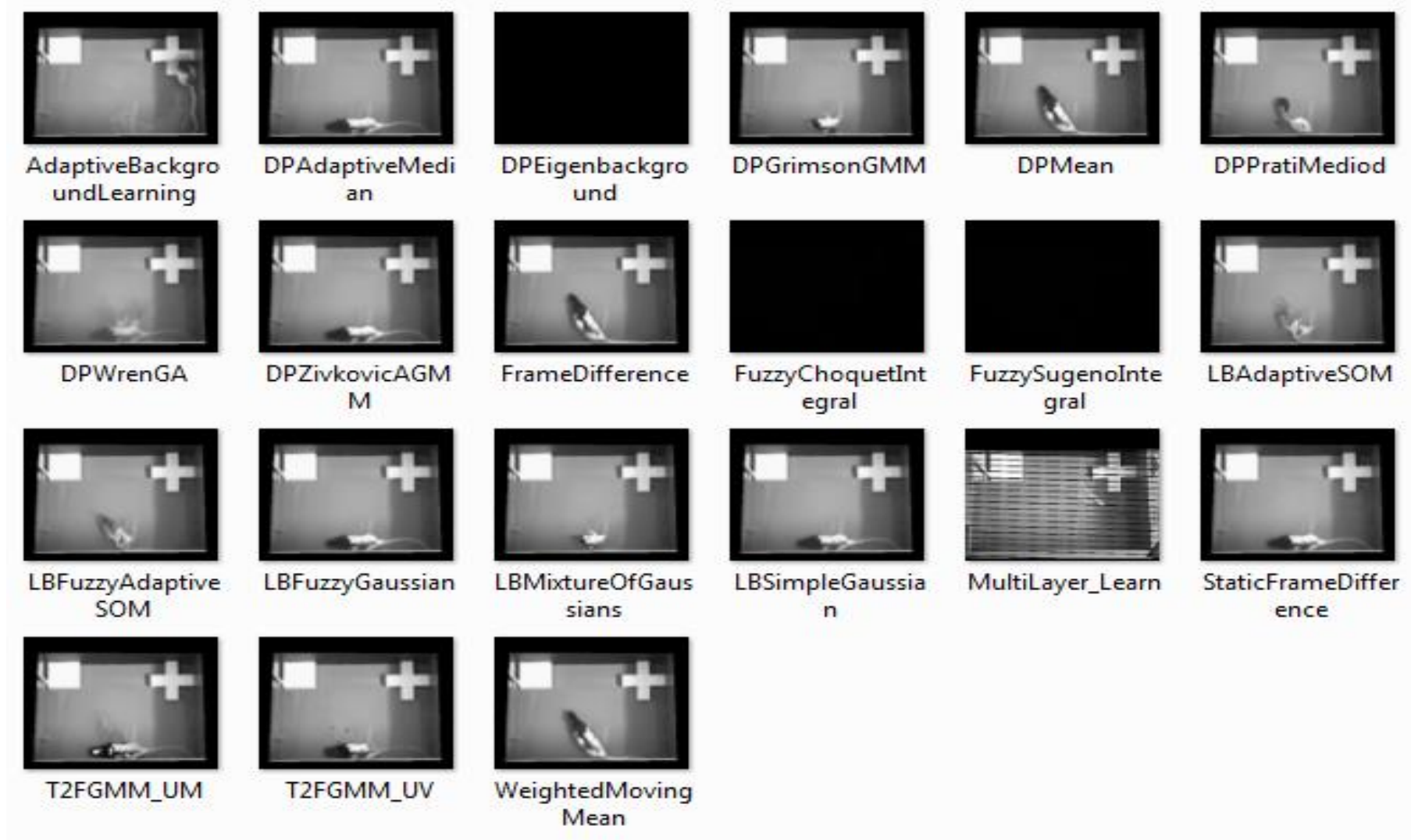


Figure 2. Minimum Thresholded Difference Frames for BS2 video.



Appendix IV. Multi-criteria evaluation to select the best method using BS2 video with thresholded Difference values.

Criteria: Speed = Quality				Speed (msec, 50%)				Difference (nr of pixels, 25%)				Difference_frequency (25%)				TOTAL	TOTAL
Method	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	SCORE	RANK			
AdaptiveBackgroundLearning	182,308	0,945	0,5	0,473	42	0,991	0,25	0,248	1	0,000	0,25	0,000	0,720	2			
DPAdaptiveMedian	184,703	0,939	0,5	0,470	1458	0,679	0,25	0,170	7	0,008	0,25	0,002	0,641	7			
DPGrimsonGMM	284,060	0,680	0,5	0,340	1477	0,675	0,25	0,169	2	0,001	0,25	0,000	0,509	13			
DPMean	213,691	0,863	0,5	0,432	2364	0,479	0,25	0,120	1	0,000	0,25	0,000	0,551	11			
DPPratiMediod	288,668	0,668	0,5	0,334	1482	0,673	0,25	0,168	5	0,005	0,25	0,001	0,503	14			
DPWrenGA	204,972	0,886	0,5	0,443	1485	0,673	0,25	0,168	1	0,000	0,25	0,000	0,611	8			
DPZivkovicAGMM	210,601	0,871	0,5	0,436	1496	0,670	0,25	0,168	2	0,001	0,25	0,000	0,604	9			
FrameDifference	164,027	0,993	0,5	0,497	879	0,806	0,25	0,202	1	0,000	0,25	0,000	0,698	3			
LBAaptiveSOM	289,133	0,666	0,5	0,333	802	0,823	0,25	0,206	1	0,000	0,25	0,000	0,539	12			
LBFuzzyAdaptiveSOM	309,041	0,614	0,5	0,307	1133	0,750	0,25	0,188	3	0,003	0,25	0,001	0,495	15			
LBFuzzyGaussian	221,139	0,844	0,5	0,422	1	1,000	0,25	0,250	5	0,005	0,25	0,001	0,673	6			
LBMixtureOfGaussians	224,250	0,836	0,5	0,418	0	1,000	0,25	0,250	768	1,000	0,25	0,250	0,918	1			
LBSimpleGaussian	201,017	0,896	0,5	0,448	69	0,985	0,25	0,246	2	0,001	0,25	0,000	0,695	4			
MultiLayer_Learn	544,312	0,000	0,5	0,000	81	0,982	0,25	0,246	1	0,000	0,25	0,000	0,246	16			
StaticFrameDifference	161,367	1,000	0,5	0,500	2938	0,353	0,25	0,088	1	0,000	0,25	0,000	0,588	10			
T2FGMM_UM	380,989	0,426	0,5	0,213	4538	0,000	0,25	0,000	1	0,000	0,25	0,000	0,213	18			
T2FGMM_UV	375,522	0,441	0,5	0,220	4136	0,089	0,25	0,022	1	0,000	0,25	0,000	0,243	17			
WeightedMovingMean	182,971	0,944	0,5	0,472	875	0,807	0,25	0,202	1	0,000	0,25	0,000	0,674	5			
Minimum	161,367				0				1								
Maximum	544,312				4538				768								

Criteria: Speed > Quality				Speed (msec, 60%)				Difference (nr of pixels, 20%)				Difference_frequency (20%)				TOTAL	TOTAL
Method	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	SCORE	RANK			
AdaptiveBackgroundLearning	182,308	0,945	0,6	0,567	42	0,991	0,2	0,198	1	0,000	0,2	0,000	0,765	2			
DPAdaptiveMedian	184,703	0,939	0,6	0,563	1458	0,679	0,2	0,136	7	0,008	0,2	0,002	0,701	7			
DPGrimsonGMM	284,060	0,680	0,6	0,408	1477	0,675	0,2	0,135	2	0,001	0,2	0,000	0,543	13			
DPMean	213,691	0,863	0,6	0,518	2364	0,479	0,2	0,096	1	0,000	0,2	0,000	0,614	11			
DPPratiMediod	288,668	0,668	0,6	0,401	1482	0,673	0,2	0,135	5	0,005	0,2	0,001	0,536	14			
DPWrenGA	204,972	0,886	0,6	0,532	1485	0,673	0,2	0,135	1	0,000	0,2	0,000	0,666	9			
DPZivkovicAGMM	210,601	0,871	0,6	0,523	1496	0,670	0,2	0,134	2	0,001	0,2	0,000	0,657	10			
FrameDifference	164,027	0,993	0,6	0,596	879	0,806	0,2	0,161	1	0,000	0,2	0,000	0,757	3			
LBAaptiveSOM	289,133	0,666	0,6	0,400	802	0,823	0,2	0,165	1	0,000	0,2	0,000	0,564	12			
LBFuzzyAdaptiveSOM	309,041	0,614	0,6	0,369	1133	0,750	0,2	0,150	3	0,003	0,2	0,001	0,519	15			
LBFuzzyGaussian	221,139	0,844	0,6	0,506	1	1,000	0,2	0,200	5	0,005	0,2	0,001	0,707	6			
LBMixtureOfGaussians	224,250	0,836	0,6	0,501	0	1,000	0,2	0,200	768	1,000	0,2	0,200	0,901	1			
LBSimpleGaussian	201,017	0,896	0,6	0,538	69	0,985	0,2	0,197	2	0,001	0,2	0,000	0,735	4			
MultiLayer_Learn	544,312	0,000	0,6	0,000	81	0,982	0,2	0,196	1	0,000	0,2	0,000	0,196	18			
StaticFrameDifference	161,367	1,000	0,6	0,600	2938	0,353	0,2	0,071	1	0,000	0,2	0,000	0,671	8			
T2FGMM_UM	380,989	0,426	0,6	0,256	4538	0,000	0,2	0,000	1	0,000	0,2	0,000	0,256	17			
T2FGMM_UV	375,522	0,441	0,6	0,264	4136	0,089	0,2	0,018	1	0,000	0,2	0,000	0,282	16			
WeightedMovingMean	182,971	0,944	0,6	0,566	875	0,807	0,2	0,161	1	0,000	0,2	0,000	0,728	5			
Minimum	161,367				0				1								
Maximum	544,312				4538				768								

Criteria: Speed < Quality	Speed (msec, 40%)				Difference (nr of pixels, 30%)				Difference_frequency (30%)				TOTAL	TOTAL
Method	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	SCORE	RANK
AdaptiveBackgroundLearning	182,308	0,945	0,4	0,378	42	0,991	0,3	0,297	1	0,000	0,3	0,000	0,675	2
DPAdaptiveMedian	184,703	0,939	0,4	0,376	1458	0,679	0,3	0,204	7	0,008	0,3	0,002	0,582	7
DPGrimsonGMM	284,060	0,680	0,4	0,272	1477	0,675	0,3	0,202	2	0,001	0,3	0,000	0,475	13
DPMean	213,691	0,863	0,4	0,345	2364	0,479	0,3	0,144	1	0,000	0,3	0,000	0,489	12
DPPratiMediod	288,668	0,668	0,4	0,267	1482	0,673	0,3	0,202	5	0,005	0,3	0,002	0,471	15
DPWrenGA	204,972	0,886	0,4	0,354	1485	0,673	0,3	0,202	1	0,000	0,3	0,000	0,556	8
DPZivkovicAGMM	210,601	0,871	0,4	0,349	1496	0,670	0,3	0,201	2	0,001	0,3	0,000	0,550	9
FrameDifference	164,027	0,993	0,4	0,397	879	0,806	0,3	0,242	1	0,000	0,3	0,000	0,639	4
LBAadaptiveSOM	289,133	0,666	0,4	0,267	802	0,823	0,3	0,247	1	0,000	0,3	0,000	0,514	10
LBFuzzyAdaptiveSOM	309,041	0,614	0,4	0,246	1133	0,750	0,3	0,225	3	0,003	0,3	0,001	0,472	14
LBFuzzyGaussian	221,139	0,844	0,4	0,338	1	1,000	0,3	0,300	5	0,005	0,3	0,002	0,639	5
LBMixtureOfGaussians	224,250	0,836	0,4	0,334	0	1,000	0,3	0,300	768	1,000	0,3	0,300	0,934	1
LBSimpleGaussian	201,017	0,896	0,4	0,359	69	0,985	0,3	0,295	2	0,001	0,3	0,000	0,654	3
MultiLayer_Learn	544,312	0,000	0,4	0,000	81	0,982	0,3	0,295	1	0,000	0,3	0,000	0,295	16
StaticFrameDifference	161,367	1,000	0,4	0,400	2938	0,353	0,3	0,106	1	0,000	0,3	0,000	0,506	11
T2FGMM_UM	380,989	0,426	0,4	0,171	4538	0,000	0,3	0,000	1	0,000	0,3	0,000	0,171	18
T2FGMM_UV	375,522	0,441	0,4	0,176	4136	0,089	0,3	0,027	1	0,000	0,3	0,000	0,203	17
WeightedMovingMean	182,971	0,944	0,4	0,377	875	0,807	0,3	0,242	1	0,000	0,3	0,000	0,620	6
Minimum	161,367				0				1					
Maximum	544,312				4538				768					

Appendix V. Frames corresponding to quality measures for different background extraction methods using Video1.

Figure 1. Minimum Difference Frames for Video1 video

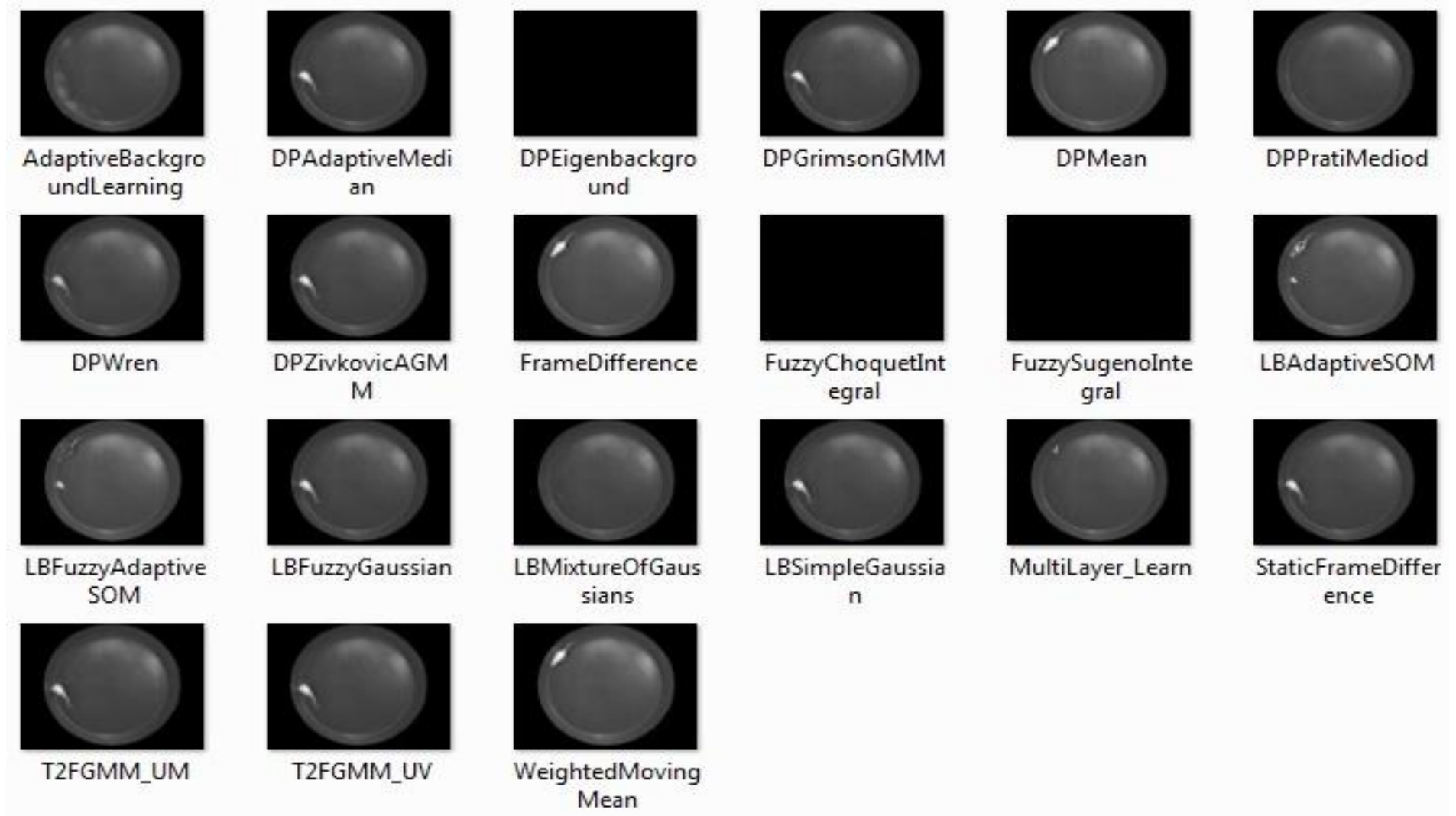
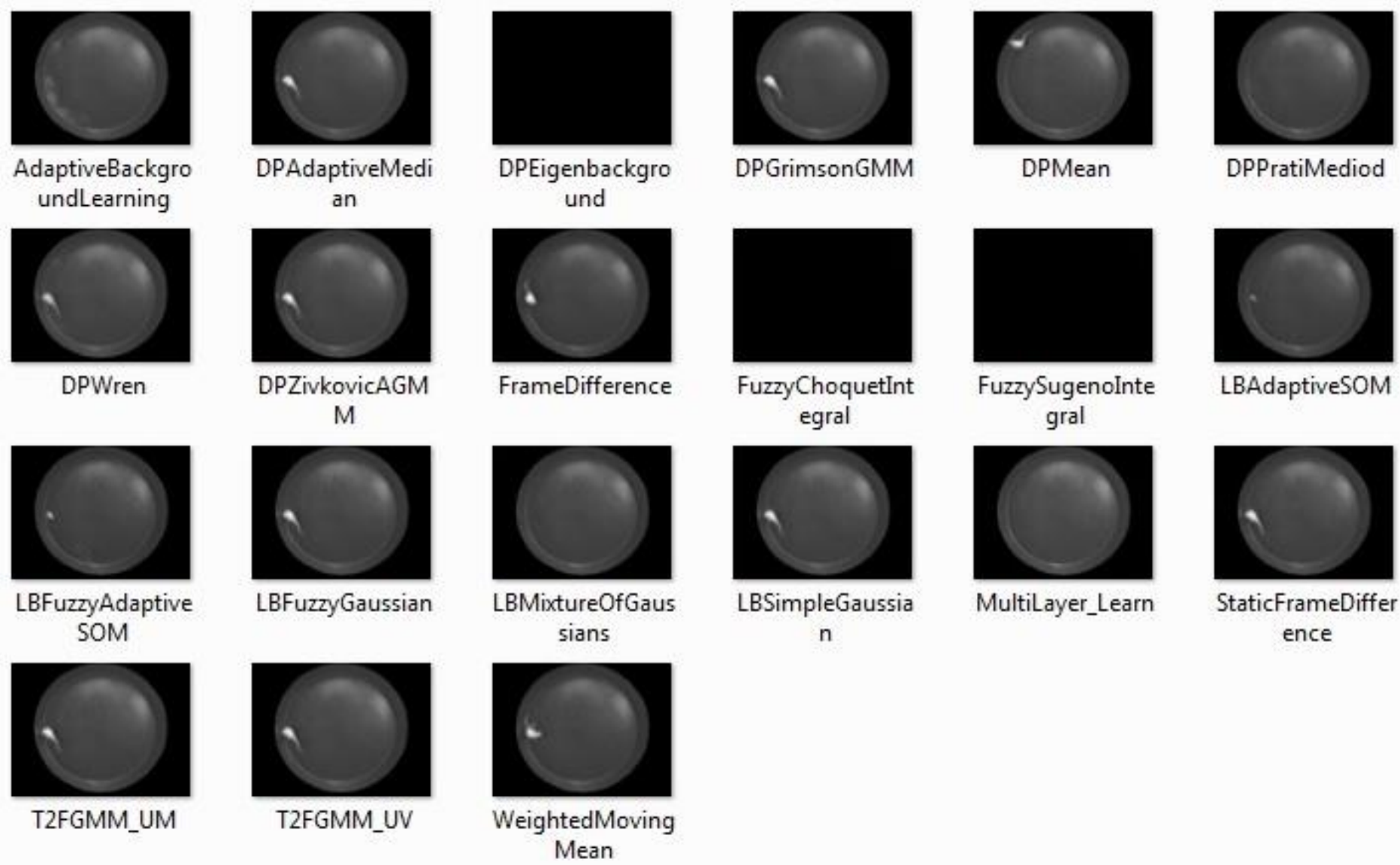


Figure 2. Minimum thresholded Difference Frames for Video1 video



Appendix VI. Multi-criteria evaluation to select the best method using Video1 video with thresholded Difference values.




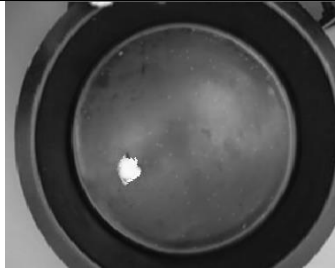


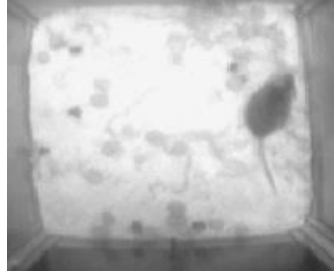
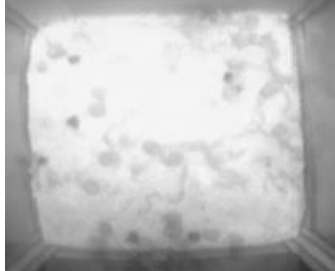


Criteria: Speed = Quality					Speed (msec, 50%)				Difference (nr of pixels, 25%)				Difference_frequency (25%)				TOTAL	TOTAL
Method	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	SCORE	RANK				
AdaptiveBackgroundLearning	6,750	0,987	0,5	0,494	158	0,826	0,25	0,206	1	0,000	0,25	0,000	0,700	2				
DPAadaptiveMedian	7,060	0,925	0,5	0,463	872	0,038	0,25	0,009	7	0,667	0,25	0,167	0,639	3				
DPGrimsonGMM	7,553	0,826	0,5	0,413	901	0,006	0,25	0,001	2	0,111	0,25	0,028	0,442	14				
DPMean	7,327	0,871	0,5	0,436	826	0,088	0,25	0,022	1	0,000	0,25	0,000	0,458	12				
DPPratiMediod	9,960	0,342	0,5	0,171	473	0,478	0,25	0,119	5	0,444	0,25	0,111	0,402	16				
DPWrenGA	7,265	0,884	0,5	0,442	851	0,061	0,25	0,015	2	0,111	0,25	0,028	0,485	11				
DPZivkovicAGMM	7,199	0,897	0,5	0,449	900	0,007	0,25	0,002	1	0,000	0,25	0,000	0,450	13				
FrameDifference	7,255	0,886	0,5	0,443	353	0,610	0,25	0,153	1	0,000	0,25	0,000	0,595	6				
LBAadaptiveSOM	7,852	0,766	0,5	0,383	38	0,958	0,25	0,240	1	0,000	0,25	0,000	0,622	5				
LBFuzzyAdaptiveSOM	8,026	0,731	0,5	0,365	90	0,901	0,25	0,225	1	0,000	0,25	0,000	0,591	7				
LBFuzzyGaussian	7,473	0,842	0,5	0,421	423	0,533	0,25	0,133	1	0,000	0,25	0,000	0,554	10				
LBMixtureOfGaussians	7,286	0,879	0,5	0,440	0	1,000	0,25	0,250	10	1,000	0,25	0,250	0,940	1				
LBSimpleGaussian	7,188	0,899	0,5	0,450	425	0,531	0,25	0,133	1	0,000	0,25	0,000	0,582	8				
MultiLayer_Learn	11,660	0,000	0,5	0,000	11	0,988	0,25	0,247	1	0,000	0,25	0,000	0,247	18				
StaticFrameDifference	6,686	1,000	0,5	0,500	429	0,526	0,25	0,132	1	0,000	0,25	0,000	0,632	4				
T2FGMM_UM	7,805	0,775	0,5	0,388	887	0,021	0,25	0,005	1	0,000	0,25	0,000	0,393	17				
T2FGMM_UV	7,544	0,828	0,5	0,414	906	0,000	0,25	0,000	2	0,111	0,25	0,028	0,442	15				
WeightedMovingMean	7,373	0,862	0,5	0,431	377	0,584	0,25	0,146	1	0,000	0,25	0,000	0,577	9				
Minimum	6,686				0				1									
Maximum	11,660				906				10									


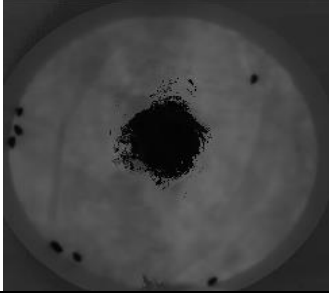




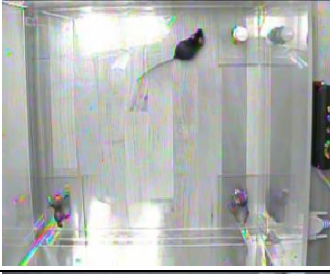

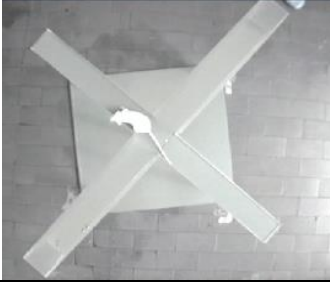
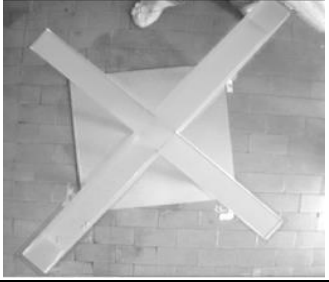
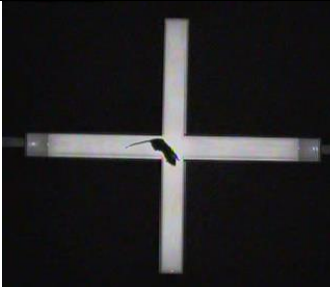
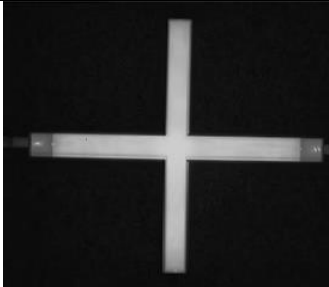
Criteria: Speed > Quality					Speed (msec, 60%)				Difference (nr of pixels, 20%)				Difference_frequency (20%)				TOTAL	TOTAL
Method	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	SCORE	RANK				
AdaptiveBackgroundLearning	6,750	0,987	0,6	0,592	158	0,826	0,2	0,165	1	0,000	0,2	0,000	0,757	2				
DPAadaptiveMedian	7,060	0,925	0,6	0,555	872	0,038	0,2	0,008	7	0,667	0,2	0,133	0,696	4				
DPGrimsonGMM	7,553	0,826	0,6	0,495	901	0,006	0,2	0,001	2	0,111	0,2	0,022	0,519	15				
DPMean	7,327	0,871	0,6	0,523	826	0,088	0,2	0,018	1	0,000	0,2	0,000	0,540	12				
DPPratiMediod	9,960	0,342	0,6	0,205	473	0,478	0,2	0,096	5	0,444	0,2	0,089	0,390	17				
DPWrenGA	7,265	0,884	0,6	0,530	851	0,061	0,2	0,012	2	0,111	0,2	0,022	0,565	11				
DPZivkovicAGMM	7,199	0,897	0,6	0,538	900	0,007	0,2	0,001	1	0,000	0,2	0,000	0,540	13				
FrameDifference	7,255	0,886	0,6	0,531	353	0,610	0,2	0,122	1	0,000	0,2	0,000	0,653	5				
LBAadaptiveSOM	7,852	0,766	0,6	0,459	38	0,958	0,2	0,192	1	0,000	0,2	0,000	0,651	6				
LBFuzzyAdaptiveSOM	8,026	0,731	0,6	0,438	90	0,901	0,2	0,180	1	0,000	0,2	0,000	0,619	9				
LBFuzzyGaussian	7,473	0,842	0,6	0,505	423	0,533	0,2	0,107	1	0,000	0,2	0,000	0,612	10				
LBMixtureOfGaussians	7,286	0,879	0,6	0,528	0	1,000	0,2	0,200	10	1,000	0,2	0,200	0,928	1				
LBSimpleGaussian	7,188	0,899	0,6	0,539	425	0,531	0,2	0,106	1	0,000	0,2	0,000	0,646	7				
MultiLayer_Learn	11,660	0,000	0,6	0,000	11	0,988	0,2	0,198	1	0,000	0,2	0,000	0,198	18				
StaticFrameDifference	6,686	1,000	0,6	0,600	429	0,526	0,2	0,105	1	0,000	0,2	0,000	0,705	3				
T2FGMM_UM	7,805	0,775	0,6	0,465	887	0,021	0,2	0,004	1	0,000	0,2	0,000	0,469	16				
T2FGMM_UV	7,544	0,828	0,6	0,497	906	0,000	0,2	0,000	2	0,111	0,2	0,022	0,519	14				
WeightedMovingMean	7,373	0,862	0,6	0,517	377	0,584	0,2	0,117	1	0,000	0,2	0,000	0,634	8				
Minimum	6,686				0				1									
Maximum	11,660				906				10									

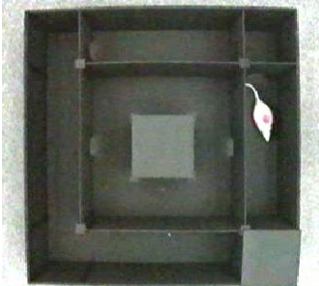
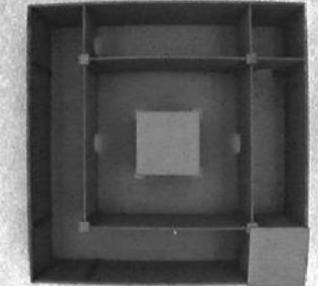
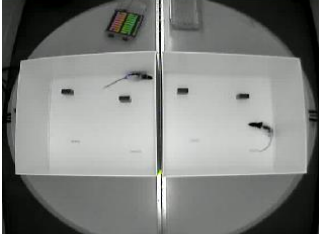


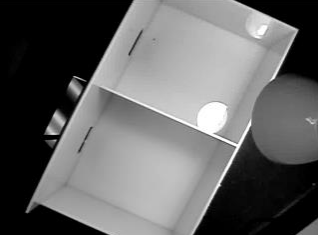




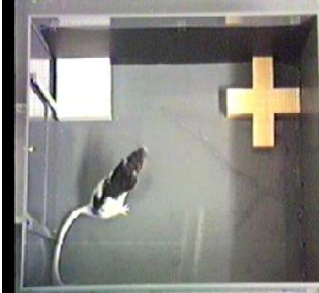
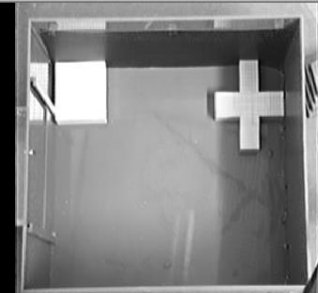
Criteria: Speed < Quality	Speed (msec, 40%)				Difference (nr of pixels, 30%)				Difference_frequency (30%)				TOTAL	TOTAL
Method	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	Val	Val (std)	Weight	Score	SCORE	RANK
AdaptiveBackgroundLearning	6,750	0,987	0,4	0,395	158	0,826	0,3	0,248	1	0,000	0,3	0,000	0,643	2
DPAdaptiveMedian	7,060	0,925	0,4	0,370	872	0,038	0,3	0,011	7	0,667	0,3	0,200	0,581	4
DPGrimsonGMM	7,553	0,826	0,4	0,330	901	0,006	0,3	0,002	2	0,111	0,3	0,033	0,365	14
DPMean	7,327	0,871	0,4	0,349	826	0,088	0,3	0,026	1	0,000	0,3	0,000	0,375	13
DPPratiMediod	9,960	0,342	0,4	0,137	473	0,478	0,3	0,143	5	0,444	0,3	0,133	0,413	11
DPWrenGA	7,265	0,884	0,4	0,353	851	0,061	0,3	0,018	2	0,111	0,3	0,033	0,405	12
DPZivkovicAGMM	7,199	0,897	0,4	0,359	900	0,007	0,3	0,002	1	0,000	0,3	0,000	0,361	16
FrameDifference	7,255	0,886	0,4	0,354	353	0,610	0,3	0,183	1	0,000	0,3	0,000	0,537	7
LBAadaptiveSOM	7,852	0,766	0,4	0,306	38	0,958	0,3	0,287	1	0,000	0,3	0,000	0,594	3
LBFuzzyAdaptiveSOM	8,026	0,731	0,4	0,292	90	0,901	0,3	0,270	1	0,000	0,3	0,000	0,562	5
LBFuzzyGaussian	7,473	0,842	0,4	0,337	423	0,533	0,3	0,160	1	0,000	0,3	0,000	0,497	10
LBMixtureOfGaussians	7,286	0,879	0,4	0,352	0	1,000	0,3	0,300	10	1,000	0,3	0,300	0,952	1
LBSimpleGaussian	7,188	0,899	0,4	0,360	425	0,531	0,3	0,159	1	0,000	0,3	0,000	0,519	9
MultiLayer_Learn	11,660	0,000	0,4	0,000	11	0,988	0,3	0,296	1	0,000	0,3	0,000	0,296	18
StaticFrameDifference	6,686	1,000	0,4	0,400	429	0,526	0,3	0,158	1	0,000	0,3	0,000	0,558	6
T2FGMM_UM	7,805	0,775	0,4	0,310	887	0,021	0,3	0,006	1	0,000	0,3	0,000	0,316	17
T2FGMM_UV	7,544	0,828	0,4	0,331	906	0,000	0,3	0,000	2	0,111	0,3	0,033	0,364	15
WeightedMovingMean	7,373	0,862	0,4	0,345	377	0,584	0,3	0,175	1	0,000	0,3	0,000	0,520	8
Minimum	6,686				0				1					
Maximum	11,660				906				10					

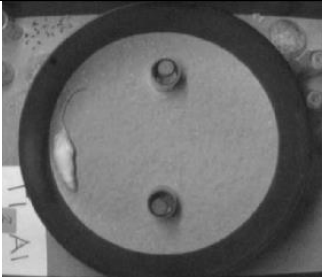
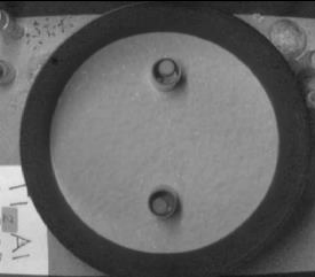

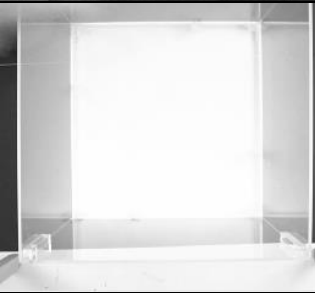


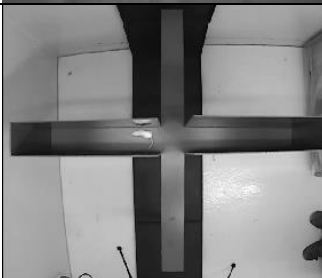
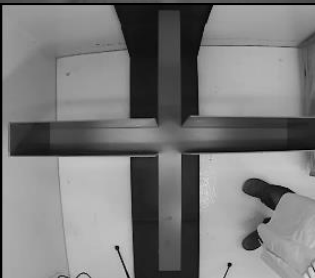



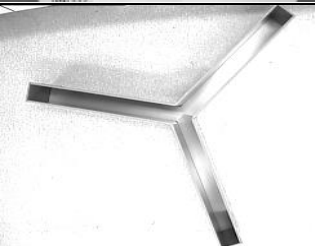
Appendix VII. Extracted backgrounds using LBMOG

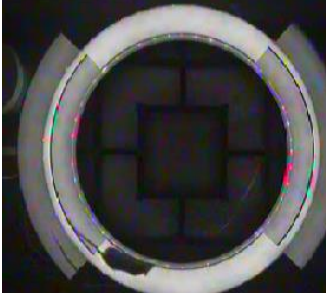
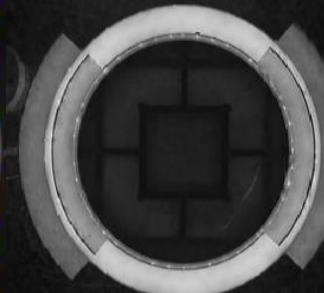

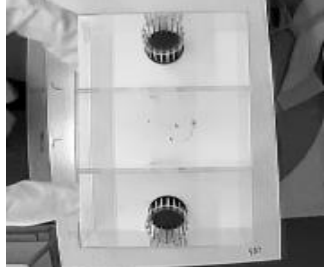
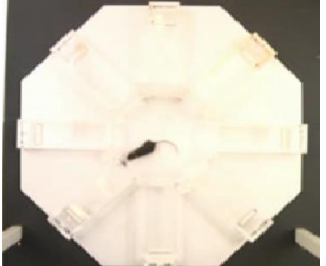
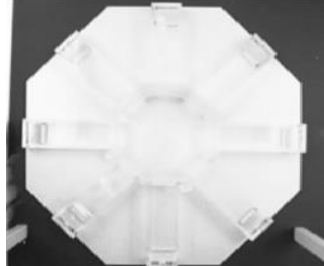






Figure 1: Results of runs with videos showing a description of the video, an image with the animal, the empty background image

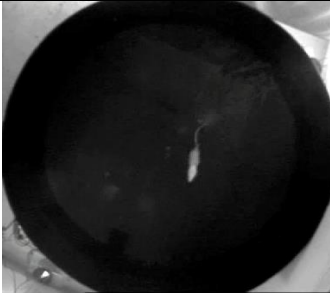



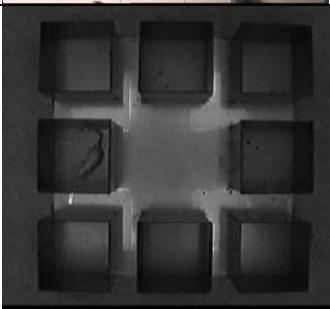
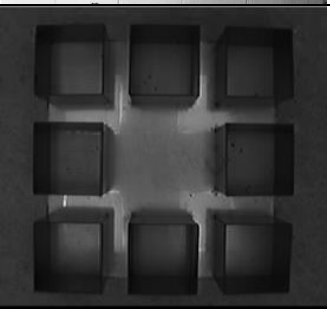

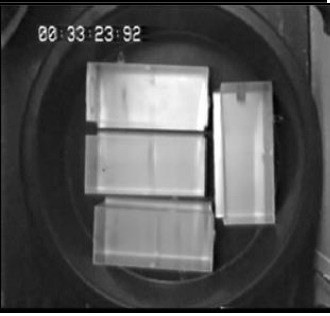
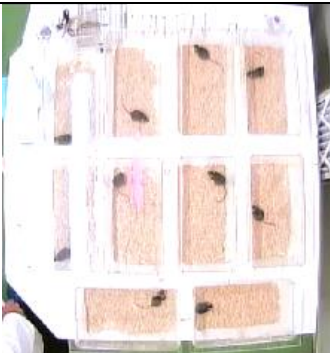
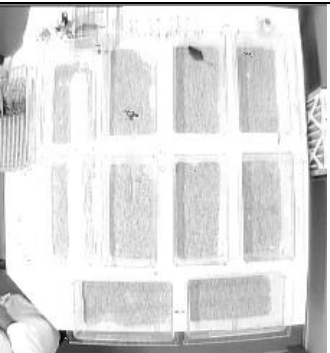
Video	Video Image with animal	Extracted background	Comments
BD1 4:18 min; <u>720 x 576;</u>			no animal at the start extracted background – good
BD2 33.4 sec; <u>352 x 288</u>			animal hardly moved extracted background – not good
BD3 10:00 min; 720 x 576			lively mouse with 2 colors extracted background – good
BD4 40:44 min; <u>352 x 288</u>			lively mouse extracted background – good
BD5 2:10 min, <u>720 x 576</u>			active big rat extracted background – good


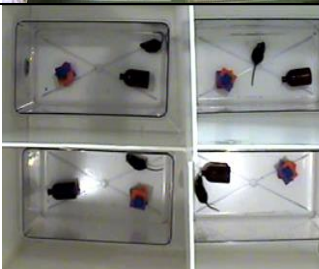
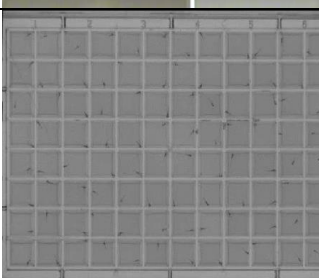
<p>BD6 <u>3:00 min</u></p>			<p>big rat with 2 colors; stayed in place extracted background – not good</p>
<p>BD7 <u>10:03 min</u></p>			<p>lively rat, second animal added later; extracted background – good</p>
<p>BD8 <u>5:00 min;</u> <u>352 x 288</u></p>			<p>plus maze extracted background – good</p>
<p>BD9 <u>10:00 min;</u> <u>352 x 288</u></p>			<p>quite some light in the arena extracted background – good</p>
<p>BD10 <u>2:09 min;</u> <u>720 x 576</u></p>			<p>no animal at the start extracted background – good</p>
<p>BD11 <u>8:55 min,</u> <u>352 x 288</u></p>			<p>no animal at the start, light turned off then on extracted background – good</p>

<p>BD12 <u>12:05 min;</u> <u>352 x 288</u></p>			<p>single rat rarely in hidden zone; no animal at start extracted background – good</p>
<p>BD13 <u>9:41 min</u></p>			<p>no animal at start; hooded rats; 2 arenas extracted background – good</p>
<p>BD14 <u>10:00 min</u></p>			<p>no animal at start extracted background – good</p>
<p>BD15 <u>6:02 min</u></p>			<p>animal at start; zebra fish extracted background – good</p>
<p>BS1 <u>12.03 min;</u> <u>720 x 576</u></p>			<p>animal at start; grid; different “floor” lighting extracted background – good</p>
<p>BS2 <u>3:14 min;</u> <u>352 x 288</u></p>			<p>no animal at start; hooded rat extracted background – good</p>

<p>BS3 <u>1:53 min;</u> <u>720 x 576</u></p>			<p>no animal at start extracted background – good</p>
<p>BS4 <u>1:19 min;</u> <u>720 x 576</u></p>			<p>no animal at start; with reflections extracted background – good</p>
<p>BS5 <u>10:00 min;</u> <u>720 x 576</u></p>			<p>animal at start extracted background – good</p>
<p>BS6 <u>4:60 min</u></p>			<p>no animal at start; elevated plus maze extracted background – good</p>
<p>BS7 <u>2:02 min</u></p>			<p>no animal at start; with holes; barnes maze extracted background – good</p>
<p>BS8 <u>2:01 min</u></p>			<p>animal at start; Y maze extracted background – good</p>

<p>BS9 <u>5:07 min;</u> <u>352 x 288</u></p>			<p>animal at start; Zero maze with backlight extracted background – good</p>
<p>BS10 <u>5:02 min</u></p>			<p>animal at start; “sociability maze” extracted background – good</p>
<p>BS12 <u>1:03 min;</u> <u>352 x 288</u></p>			<p>no animal at start; radial maze extracted background – good</p>
<p>BS13 <u>10:04 min</u></p>			<p>animal at start; hole board extracted background – good</p>
<p>BS14 <u>10:05 min;</u> <u>352 x 288</u></p>			<p>animal at start; 2 petri dishes extracted background – good</p>
<p>BS15 <u>7:40 min;</u> <u>352 x 288</u></p>			<p>no animal at start; water maze (small ripples when moved) extracted background – good</p>

<p>BS16 <u>1:00 min</u></p>			<p>no animal at start; water maze (small ripples when moved) extracted background – good</p>
<p>BM1 <u>4:35 min;</u> <u>720 x 576</u></p>			<p>no animal at start; 6 arenas, 5 filled extracted background – good</p>
<p>BM2 <u>14:31 min;</u> <u>352 x 288</u></p>			<p>animal at start; only one arena used extracted background – good</p>
<p>BM3 <u>18:12 min;</u> <u>352 x 288</u></p>			<p>no animal at start; 4 boxes rats inserted by user with shadows; second trial after 50 mins. extracted background – good</p>
<p>BM4 <u>2:50:06 hr</u></p>			<p>no animal at start; 10 arenas, all animal were in after 10.06.44; at this time one or 2 did not move anymore extracted background – good except for 2</p>

<p>BM5 <u>3:00:20 hr</u></p>		<p>no animal at start; all animals in at 9.24.6; some arenas displaced extracted background – good except for 3</p>
<p>BM6 <u>22.5 sec</u></p>		<p>animals at start; 4 arenas extracted background – good</p>
<p>BS11 <u>0:30 min;</u> <u>640 x 480</u></p>		<p>animals at start; well plate; 96 arenas filled with fish extracted background – good</p>

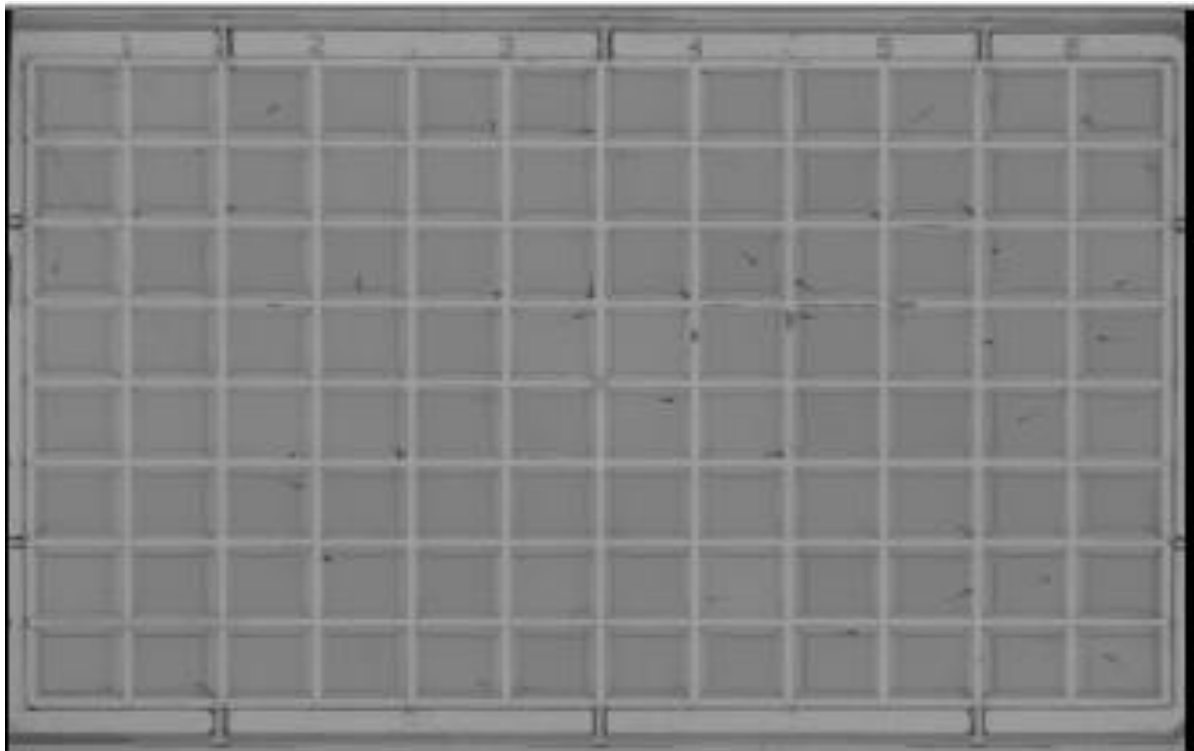


Figure 2. Result for 96 arenas (did not wait until all arenas were emptied).